



# **FPGA Implementations for Parallel Multidimensional Filtering Algorithms**

By

*Sami Kadhim Hasan*

A thesis submitted to the School of Electrical and Electronic Engineering  
in partial fulfilment of the requirements for the degree of  
Doctor of Philosophy

Faculty of Science, Agriculture and Engineering  
Newcastle University,

June 2013

## ABSTRACT

One and multi dimensional raw data collections introduce noise and artifacts, which need to be recovered from degradations by an automated filtering system before, further machine analysis. The need for automating wide-ranged filtering applications necessitates the design of generic filtering architectures, together with the development of multidimensional and extensive convolution operators. Consequently, the aim of this thesis is to investigate the problem of automated construction of a generic parallel filtering system. Serving this goal, performance-efficient FPGA implementation architectures are developed to realize parallel one/multi-dimensional filtering algorithms. The proposed generic architectures provide a mechanism for fast FPGA prototyping of high performance computations to obtain efficiently implemented performance indices of area, speed, dynamic power, throughput and computation rates, as a complete package. These parallel filtering algorithms and their automated generic architectures tackle the major bottlenecks and limitations of existing multiprocessor systems in wordlength, input data segmentation, boundary conditions as well as inter-processor communications, in order to support high data throughput real-time applications of low-power architectures using a Xilinx Virtex-6 FPGA board.

For one-dimensional raw signal filtering case, mathematical model and architectural development of the generalized parallel 1-D filtering algorithms are presented using the 1-D block filtering method. Five generic architectures are implemented on a Virtex-6 ML605 board, evaluated and compared. A complete set of results on area, speed, power, throughput and computation rates are obtained and discussed as performance indices for the 1-D convolution architectures. A successful application of parallel 1-D cross-correlation is demonstrated.

For two dimensional greyscale/colour image processing cases, new parallel 2-D/3-D filtering algorithms are presented and mathematically modelled using input decimation and output image reconstruction by interpolation. Ten generic architectures are implemented on the Virtex-6 ML605 board, evaluated and compared. Key results on area, speed, power, throughput and computation rate are obtained and discussed as performance indices for the 2-D convolution architectures. 2-D image reconfigurable processors are developed and implemented using single, dual and quad MAC FIR units. 3-D Colour image processors are devised to act as 3-D colour filtering engines. A 2-D cross-correlator parallel engine is successfully developed as a parallel 2-D matched filtering algorithm for locating any MRI slice within a MRI data stack library. Twelve 3-D MRI filtering operators are plugged in and adapted to be suitable for biomedical imaging, including 3-D edge operators and 3-D noise smoothing operators.

Since three dimensional greyscale/colour volumetric image applications are computationally intensive, a new parallel 3-D/4-D filtering algorithm is presented and mathematically modelled using volumetric data image segmentation by decimation and output reconstruction by interpolation, after simultaneously and independently performing 3-D filtering. Eight generic architectures are developed and implemented on the Virtex-6 board, including 3-D spatial and FFT convolution architectures. Fourteen 3-D MRI filtering operators are plugged and adapted for this particular biomedical imaging application, including 3-D edge operators and 3-D noise smoothing operators. Three successful applications are presented in 4-D colour MRI (fMRI) filtering processors, k-space MRI volume data filter and 3-D cross-correlator.

*To Ali Ibn Abi Talib*

*For his highly inspirational attitudes*

*And My Family*

*For, their support,*

*Encouragement*

*And patient all the way*

## ACKNOWLEDGEMENTS

I wish to express my eternal praise to the mighty ALLAH who empowers me with mature patient and young strength to complete this research project.

Thanks to IRAQ. It would not have been possible to achieve this doctoral research project without the kind sponsorship of IRAQI government, represented by his high Excellency the prime minister “Nuri al-Maliki”, the ministry of higher education and scientific research Prof. “Ali al-Adeeb”, the president of Al-Nahrain University Prof. “M. Jabber” as well as the Cultural Attaché prof. “A. al-Baghdadi”. My thanks and gratitude to the devoted crew of the cultural department in the IRAQI embassy in LONDON.




I owe my sincere appreciation and deepest respect to my great supervisors Prof. Alex Yakovlev and Prof. Said Boussakta for their sincere support, helpful advice and honest guidance throughout this research project.

Prof. Alex Yakovlev provided an ideal framework for this project. His excellent guidance in the modern methods of low-power hardware research, encouragement, endless patience, and amazing sense of academic leadership has been inspirational and essential for the accomplishment of this work.

Prof. Said Boussakta first introduced me to the fascinating field of parallel multi-dimensional convolutions/correlations algorithms research field. I was so impressed by his compelling enthusiasm and sincere interest in Digital signal processing algorithms, that, before long, I found myself working with him, not only in the algorithmic research field, but also in the massively parallel architectural research field.

## Abbreviations and Symbols

A list of the main abbreviations and Symbols that used in this thesis is given below; all their definitions are given within the text:

Symbol	Detail
	One dimension point-by-point complex multiplication
	Two dimensions point-by-point complex multiplication
	Three dimensions point-by-point complex multiplication
*	One dimension linear convolution
**	Two dimensions linear convolution
***	Three dimensions linear convolution
1-D	One dimension
2-D	Two dimensions
3-D	Three dimensions
4-D	Four dimensions
ASIC	Application Specific Integrated Circuit
FPGA	Field Programmable Gate Array
VLSI	Very Large Scale Integration
BRAM	Block random Access Memory; RAMB 18E1s (slice)
DSP	DSP 48E1s (slices)
DSPs	DSP processors
GAs	Gate Arrays
FIR	Finite impulse response filter
$h(m)$	FIR filter 1-D impulse response, 1-D kernel
$h(m_1, m_2)$	FIR filter 2-D impulse response, 2-D kernel
$h(m_1, m_2, m_3)$	FIR filter 3-D impulse response, 3-D kernel
NMNT	New Mersenne number transform
ns	Nanosecond
ms	Millisecond
MRI	Magnetic Resonance Imaging
fMRI	functional Magnetic Resonance Imaging
$x(n)$	Original audio signal
$x(n_1, n_2)$	Original 2-D input image
$\mathbf{x}(n_1, n_2)$	Original 3-D colour input image
$x(n_1, n_2, n_3)$	Input 3-D volume data image
$\mathbf{x}(n_1, n_2, n_3)$	Input 4-D colour volume data image
$y(n)$	Filtered Output audio signal
$y(n_1, n_2)$	Filtered output 2-D image
$\mathbf{y}(n_1, n_2)$	Filtered output colour 3-D image
$y(n_1, n_2, n_3)$	Filtered output 3-D image
$\mathbf{y}(n_1, n_2, n_3)$	Filtered output 4-D colour volume data image
1-D MAC	One dimension Multiply/Accumulate DSP operation
2-D MAC	Two dimension Multiply/Accumulate DSP operation
3-D MAC	Three dimension Multiply/Accumulate DSP operation
mW	Milli Watts
MHz	Mega Hertz

Symbol	Detail
BPS	Block Per Second, 1-D throughput unit
FPS	Frame Per Second, 2-D throughput unit
CFPS	Colour Frame Per Second, 3-D throughput unit
VPS	Volume Per Second, 3-D throughput unit
CVPS	Colour Volume Per Second, 4-D throughput unit
GMACPS	Giga MAC Per Second, computation rate unit
1-D FFT	One Dimension Fast Fourier Transform
2-D FFT	Two Dimensions Fast Fourier Transform
3-D FFT	Three Dimensions Fast Fourier Transform
RCFFFT	row-column-frame FFT
1-D IFFT	One Dimension Inverse Fast Fourier Transform
2-D IFFT	Two Dimension Inverse Fast Fourier Transform
3-D IFFT	Three Dimension Inverse Fast Fourier Transform
XST	Xilinx Synthesis Tool
I/O	Input/Output
XSG	Xilinx System Generator for DSP
ISE	Integrated System Environment (Xilinx)
HDL	Hardware Design Language
RCF	Row-Column-Frame implementation
SDK	System Design Kit (Xilinx)
EDIF	Electronic Design Interchange Format
pixel	2-D Picture element
voxel	3-D volume element
SRL	Shift Register

## Table of Contents

ABSTRACT .....	ii
ACKNOWLEDGEMENTS .....	iv
Abbreviations and Symbols .....	v
List of Figures .....	xi
List of Tables.....	xiii
List of Equations .....	xv
<b>Chapter 1. INTRODUCTION .....</b>	<b>1</b>
1.1 Background .....	1
1.2 Related Research Works .....	2
1.2.1 Parallel Field Programming Tools .....	2
1.2.2 Parallel 1-D Signal Filtering Architectures.....	4
1.2.3 Parallel 2-D Image Filtering Architectures .....	5
1.2.4 Parallel 3-D Volumetric Data Image Filtering Architectures .....	8
1.3 Research Aims and Objectives.....	9
1.3.1 Aims .....	10
1.3.2 Objectives.....	11
1.4 Contributions per Chapter .....	11
1.4.1 Contributions of Chapter 3.....	11
1.4.2 Contributions of Chapter 4.....	11
1.4.3 Contributions of Chapter 5.....	12
1.5 Organization of the Thesis .....	12
<b>Chapter 2. Parallelism Analysis in Filtering Algorithms and FPGA-Based Architectures .....</b>	<b>14</b>
2.1 Introduction .....	14
2.2 Parallel Filtering Algorithms.....	15
2.2.1 Temporal Parallelism .....	16
2.2.2 Spatial Parallelism.....	16
2.2.3 Logical Parallelism .....	17
2.3 FPGA-Based Implementation of Parallel Filtering Algorithms.....	18
2.4 Parallel Digital Filtering using FPGAs .....	19
2.5 Overall FPGA Implementation Steps.....	21
2.6 Parallel Field Programming using Xilinx System Generator Tool .....	22
2.7 Performance Indices.....	25
2.7.1 Logic Area Occupation .....	25
2.7.2 Dynamic Power Consumption .....	27
2.7.3 Clock Speed .....	27
2.7.4 Total Throughput .....	27
2.7.5 Computation Rate .....	28
2.8 Conclusion.....	28
<b>Chapter 3. Parallel 1-D Filtering Algorithm and its FPGA Implementations .....</b>	<b>29</b>
3.1 Introduction .....	29
3.2 Research Concepts Definitions .....	30
3.2.1 Theory of Linear Block Filtering Method.....	30
3.2.2 Linear-Phase 1-D FIR Digital Filter .....	31
3.2.3 1-D FFT Convolver Engine .....	31
3.2.4 Total Throughput .....	32
3.2.5 Total Computation Rate .....	33
3.3 The Generalized Parallel 1-D Linear Block Filtering Algorithm .....	34

3.3.1 Input Segmentation Stage .....	34
3.3.2 Parallel Filtering Stage .....	35
3.3.3 Output Reconstruction Stage .....	36
3.4 Parallel 1-D Temporal Convolver Architectures .....	36
3.4.1 Parallel Single MAC Convolver Architecture .....	38
3.4.2 Parallel Dual MAC Convolver Architecture .....	39
3.4.3 Parallel Quad MAC Convolver Architecture .....	40
3.4.4 Performance Indices of the Three Parallel Multi-MAC Convolver Architectures .....	41
3.5 Parallel 1-D FFT Convolver Architectures .....	44
3.5.1 Parallel 1-D FFT Convolver Architecture .....	45
3.5.2 Fast Parallel 1-D FFT Convolver Architecture .....	46
3.5.3 Performance Indices of Fast Parallel 1-D FFT Filtering Architecture .....	48
3.6 1-D Cross-Correlation Application: FPGA Architecture for Parallel 1-D Matched Filtering Algorithm .....	50
3.7 Conclusion .....	51
<b>Chapter 4. Parallel 2-D Greyscale/Colour Image Filtering Algorithm and Its FPGA Implementations .....</b>	<b>52</b>
4.1 Introduction .....	52
4.2 Research Concepts Definitions .....	53
4.2.1 Linear 2-D Stream Filtering Method .....	53
4.2.2 Linear 2-D Filters .....	54
4.2.3 Spatiotemporal Convolver Engine .....	56
4.2.4 2-D FFT Convolver Engine .....	58
4.2.5 Total 2-D Throughput .....	59
4.2.6 Total Computation Rate .....	60
4.3 The Generalized Parallel 2-D Linear Image Filtering Algorithm .....	61
4.3.1 Input Decimation by 2 for the 2-D Image Stream Filtering .....	61
4.3.2 Parallel 2-D Filtering Stage .....	62
4.3.3 Output Interpolation by 2 Reconstruction Stage .....	62
4.4 Parallel 2-D Spatiotemporal Convolver Architectures .....	63
4.4.1 Parallel 2-D Single MAC Convolver Architectures .....	65
4.4.2 Parallel 2-D Dual MAC Convolver Architectures .....	67
4.4.3 Parallel 2-D Quad MAC Convolver Architectures .....	67
4.4.4 Performance Indices of the Parallel Spatiotemporal Convolver Architectures .....	68
4.5 Three Dimensions Application: Parallel Colour MRI Filtering .....	71
4.6 FPGA Implementation as Indirect Parallel 2-D FFT Filtering Architectures .....	74
4.6.1 Single 2-D FFT Filtering Unit Implementation .....	75
4.6.2 Single 2-D IFFT Convolution k-Space MRI Unit Implementation .....	77
4.6.3 Fast Parallel 2-D FFT Filtering Architecture .....	78
4.6.4 Performance Indices of Parallel 2-D FFT Convolution Architectures .....	79
4.7 Cross-correlator application: FPGA Architecture for Parallel 2-D MRI Matched Filtering Algorithm .....	83
4.8 Conclusion .....	85
<b>Chapter 5. Parallel 3-D Greyscale/Colour Image Filtering Algorithm and Its FPGA Implementations .....</b>	<b>86</b>
5.1 Introduction .....	86
5.2 Research Concepts Definitions .....	87
5.2.1 Linear 3-D Stream Filtering Method .....	87



5.2.2 Linear 3-D FIR Filters.....	88
5.2.3 3-D spatial Convolver Engine.....	91
5.2.4 3-D FFT Convolver Engine .....	92
5.2.5 Parallel 3-D Convolver Architectures Throughput.....	94
5.2.6 Total Computation Rate .....	94
5.3 The Generalized Parallel 3-D Linear Image Filtering Algorithm.....	95
5.3.1 Input Decimation by 2 for the 3-D Image Stream Filtering.....	96
5.3.2 Parallel 3-D Filtering Stage.....	96
5.3.3 3-D Image Reconstruction Output Stage .....	97
5.4 Parallel 3-D Spatial Convolver Architectures.....	97
5.4.1 Parallel 3-D Single MAC Convolver Filter Engines Architectures .....	99
5.4.2 Parallel 3-D Dual MAC Convolver Filter Engines Architectures .....	101
5.4.3 Parallel (2×8×3) Quad MAC 2-D Convolver Filter Engine Architecture ...	102
5.4.4 Performance Indices of Parallel 3-D Spatial Convolver Architectures .....	103
5.5 Four Dimension (4-D) Application: fMRI or 4-D Colour MRI Volume Filtering .....	108
5.6 FPGA Implementation as Parallel 3-D FFT Convolver Architectures .....	113
5.6.1 Single 3-D FFT Convolver Unit Implementation.....	114
5.6.2 Single 3-D IFFT Convolver k-Space MRI Filtering Unit Implementation .	116
5.6.3 Fast Parallel 3-D FFT Convolution Architecture.....	118
5.6.4 Performance Indices of Parallel 3-D FFT Convolver Architectures.....	120
5.7 3-D Cross-Correlator Application: Parallel 3-D MRI Matched Filtering Algorithm .....	124
5.8 Conclusion.....	125
<b>Chapter 6. Conclusion and Future Work .....</b>	<b>126</b>
6.1 Introduction .....	126
6.2 Original Contributions .....	126
6.2.1 Parallel 1-D Signal Filtering Architectures.....	126
6.2.2 Parallel 2-D Grayscale/Colour Image Filtering Architectures.....	127
6.2.3 Parallel 3-D Grayscale/Colour Image Filtering Architectures.....	128
6.3 Overall Conclusions .....	128
6.3.1 Performance Indices of the Parallel Multi-Dimensional Architectures .....	129
6.3.2 Parallel 1-D Signal Filtering Architectures.....	129
6.3.3 Parallel 2-D Grayscale/Colour Image Filtering Architectures.....	130
6.3.4 Parallel 3-D Grayscale/Colour Image Filtering Architectures.....	131
6.4 Recommendation for Future Work .....	132
<b>APPENDICES .....</b>	<b>134</b>
<b>Appendix A. Architecture 17's first filtering stage out of eight.....</b>	<b>134</b>
<b>Appendix B. Logic area occupation of architecture 17 mapped and shown graphically using Xilinx FPGA Editor Tool .....</b>	<b>135</b>
<b>Appendix C. Architecture 17's RTL schematic diagram.....</b>	<b>136</b>
List of Publications .....	137
[2] S. Hasan, S. Boussakta, and A. Yakovlev, "High Performance Implementations of Parallel 3-D fMRI Filtering Algorithm using a Class of Spatial Convolution Architectures," Paper in preparation .. <b>Error! Bookmark not defined.</b>	
[3] S. Hasan, S. Boussakta, and A. Yakovlev, " Performance-Aware Architectures for Parallel 4-D color fMRI Filtering Algorithm: A Complete	

Performance Indices Package," Paper in preparation	<b>Error!</b>	<b>Bookmark</b>	<b>not</b>
<b>defined.</b>			
REFERENCES.....			138

## List of Figures

Figure 2.1: A temporal parallelism is exploiting an independent convolver and networking using pipeline .....	16
Figure 2.2: Spatial parallelism is dedicating parallel convolvers for each data segment	17
Figure 2.3: Logical parallelism exploits an independent Convolver for each input segment as in (a) and a classical example is in (b) .....	18
Figure 2.4: Overall FPGA implementation flow for Parallel Multi-dimensional Data filtering Algorithm .....	22
Figure 2.5: Xilinx System Generator tool complete block sets. ....	24
Figure 3.1: 1-D FFT convolver unit .....	31
Figure 3.2: The Generalized Parallel 1-D Linear block Filtering Algorithm .....	34
Figure 3.3: Parallel 1-D multi-MAC filtering architecture .....	37
Figure 3.4: Unit 1; 1-D Single MAC convolver implementation .....	38
Figure 3.5: Unit 2; Implementation of 1-D Dual MAC convolver .....	39
Figure 3.6: Unit 3; 1-D Quad MAC convolver implementation.....	41
Figure 3.7: Architecture1's filtering results of a raw real-time speech (22050 Hz /1Ch/16 bit) signal at 1-D FIR kernels of 2 coefficients.....	43
Figure 3.8:Architecture2's filtering results of a raw real-time speech (22050 Hz /1Ch/16 bit) signal at 1-D FIR kernels of 4 coefficients.....	44
Figure 3.9:Architecture3's filtering results of a raw real-time speech (22050 Hz /1Ch/16 bit) signal at 1-D FIR kernels of 8 coefficients.....	44
Figure 3.10: Architecture 4; implementation of fast single 1-D FFT convolver unit.....	45
Figure 3.11: Implementation of the Block Buffer component.....	46
Figure 3.12: Architecture 5; implementation of Fast Parallel 1-D FFT block Filtering Algorithm in the Virtex-6 FPGA board .....	48
Figure 3.13: Architecture 5's filtering results of the noisy real-time speech (22050 Hz/ 1Ch/ 16 bit) signal at $M= 63/ N=128$ .....	50
Figure 3.14: The matched filtering results using architecture 5 at $M= 63/ N=128$ . (a) Input speech signal, (b) Target input segment, (c) Matching results. ....	51
Figure 4.1: A single digital image is partitioned into spatial parallelism of pixels for parallel digital processing. ....	53
Figure 4.2: Logic parallelism; a single spatial parallelism image block is converted into a temporal parallelism for Stream processing.....	54
Figure 4.3: Implementation of the Generic 2-D Spatiotemporal Convolver Unit .....	56
Figure 4.4: Five temporal parallelism copies of a digital image are streamed into a $5 \times 5$ FIR filter.....	57
Figure 4.5: 2-D FFT convolver unit.....	58
Figure 4.6: 2-D FFT convolver unit is separable to be implemented as 1-D FFT pair engine architecture. ....	58
Figure 4.7: The Generalized Parallel 2-D Linear Image Filtering Algorithm .....	61

Figure 4.8: Spatiotemporal implementation of the Parallel 2-D Convolution Algorithm .....	64
Figure 4.9: Unit 4; (2×2) Single MAC Convolver Architecture .....	65
Figure 4.10: Unit 5; (3×3) Single MAC Convolver Architecture .....	65
Figure 4.11: Unit 6; (5×5) Single MAC Convolver Architecture .....	66
Figure 4.12: Row buffer implementation.....	66
Figure 4.13: Unit 7; (2×4) Dual MAC convolver Architecture.....	67
Figure 4.14: Unit 8; (4×4) Dual MAC convolver Architecture.....	67
Figure 4.15: Unit 9; (2×8) Quad MAC convolver Architecture.....	68
Figure 4.16: Unit 10; (8×8) Quad MAC convolver Architecture.....	68
Figure 4.17: Architecture 13; the implementation of the Fast Single 2-D FFT Filtering unit on the Virtex-6 FPGA board.....	75
Figure 4.18: transpose unit implementation.....	76
Figure 4.19: Architecture 14; implementation of the Fast Single 2-D IFFT convolution k-space MRI Filtering unit on the Virtex-6 FPGA board .....	77
Figure 4.20: Architecture 15; implementation of Fast Parallel 2-D FFT convolution Algorithm on the Virtex-6 FPGA board .....	79
Figure 4.21: The parallel 2-D MRI matched filtering (cross-correlation) example using architecture 15 .....	85
Figure 5.1:3-D image physical representation. ....	88
Figure 5.2: Generalized 3-D Spatial Convolver Unit implementation .....	91
Figure 5.3: Fast filtering by 3-D FFT convolver unit .....	92
Figure 5.4: 3-D separable FFT convolver unit.....	93
Figure 5.5: The Generalized Parallel 3-D image Linear Filtering Algorithm.....	95
Figure 5.6: The implementation of the Parallel 3-D Spatial Convolution Algorithm on the Virtex-6 FPGA board; ML 605 development kit.....	99
Figure 5.7: Unit 7; (2×2×3) Single MAC convolver Architecture .....	100
Figure 5.8: Unit 8; (3×3×3) Single MAC convolver Architecture .....	101
Figure 5.9: Unit 9; (2×4×3) Dual MAC convolver Architecture.....	102
Figure 5.10: Unit 10; (4×4×3) Dual MAC convolver Architecture.....	102
Figure 5.11: Unit 11; (2×8×3) Quad MAC convolver Architecture .....	103
Figure 5.12: Architecture 21; the implementation of the Fast Single 3-D FFT convolution g unit in the Virtex-6 FPGA board.....	114
Figure 5.13: frame-to-frame unit implementation .....	115
Figure 5.14: Architecture 22; an implementation of Fast Single 3-D IFFT convolution k-space MRI volume data Filtering unit in the Virtex-6 FPGA board.....	117
Figure 5.15: Architecture 23; an implementation of Fast Parallel 3-D FFT convolution Algorithm in the Virtex-6 FPGA board .....	119

## List of Tables

Table 1.1: parallel signal processing algorithm implementations in FPGA/ASIC.....	3
Table 2.1: Characteristics of the Virex-6 XC6VLX240T development FPGA board....	26
Table 3.1: Logic Devices utilization by the three 1-D temporal convolver units.....	39
Table 3.2: Logic Devices utilization by the three parallel temporal convolver architectures .....	42
Table 3.3: Performance indices of the parallel 1-D multi-MAC convolver filter architectures .....	42
Table 3.4: Logic Devices utilization by the parallel 1-D FFT filtering' architectures ...	49
Table 3.5: Performance indices of the parallel 1-D FFT filtering' architectures.....	49
Table 4.1: some of the common edge and noise smoothing FIR filter of 5x5 kernels ...	55
Table 4.2: Logic Devices utilization by the 2-D multi-MAC convolver units .....	66
Table 4.3: Logic Devices utilization by each of the parallel spatiotemporal filtering architectures .....	69
Table 4.4: Performance indices of the parallel spatiotemporal filtering architectures ...	69
Table 4.5: Filtering results of 64×64 greyscale MRI using architecture 8 .....	70
Table 4.6: Filtering results for grayscale 64×64 MRI of twelve improved 2-D FIR filter operators using architecture 8 .....	71
Table 4.7: Filtering results for colour 224×224×3 MRI of twelve developed 2-D FIR filter operators using architecture 9.....	73
Table 4.8: Filtering results for colour 224×224×3 MRI using the parallel spatiotemporal filtering architectures .....	74
Table 4.9: Logic Devices utilization by architecture 13 .....	80
Table 4.10: Logic Devices utilization by architecture 15 .....	80
Table 4.11: Performance indices of architecture 13.....	81
Table 4.12: Performance indices of architecture 15.....	81
Table 4.13: the filtering results for greyscale ( $N_1 \times N_2$ ) MRI of the generic 2-D FIR Sharpen operators using architecture 13 .....	82
Table 4.14: the filtering results for greyscale ( $N_1 \times N_2$ ) MRI of the generic 2-D FIR Sharpen operators using architecture 15 .....	82
Table 4.15: Sharpening results for k-space greyscale ( $256 \times 256$ ) MRI of the generic 2-D FIR Sharpen operators using architecture 14 .....	83
Table 5.1: Fourteen generic 3-D edge and noise smoothing filter operators (kernels), where O.F is Operator Factor.....	90
Table 5.2: Logic Devices utilization by the 3-D spatial convolver units.....	100
Table 5.3: Logic Devices utilization by each of the five parallel spatial convolver architectures .....	104
Table 5.4: Performance indices of each of the 3-D MAC FIR filter architectures .....	104
Table 5.5: The filtering results of grayscale $256 \times 256 \times 20$ volume fMRI for fourteen generic 3-D FIR filter operators using architecture 17 .....	106

Table 5.6: The filtering results for grayscale $256 \times 256 \times 20$ volume fMRI of fourteen improved 2-D FIR filter operators using architecture 17 .....	107
Table 5.7: Throughput of fMRI (colour MRI) volume using architectures 16 to 20....	109
Table 5.8: Filtering results for colour $256 \times 256 \times 3 \times 4$ MRI volumetric.....	109
Table 5.9 : the 4-D filtering results for colour $256 \times 256 \times 3 \times 4$ MRI volume of five generic 3-D FIR noise smoothing filter operators using architecture 17 .....	110
Table 5.10: the 4-D filtering results for colour $256 \times 256 \times 3 \times 20$ MRI volume of nine improved 3-D FIR Edge enhancement filter operators using architecture 17 .....	111
Table 5.11: the 4-D filtering results for colour $256 \times 256 \times 3 \times 20$ MRI volume of five improved 3-D FIR noise smoothing filter operators using architecture 17 .....	112
Table 5.12: Filtering results for $256 \times 256 \times 3 \times 20$ colour MRI of the improved 3-D Edge Enhancement operators using architectures 16, 17, 18, 19 and 20. ....	113
Table 5.13: Logic Devices utilization by architecture 21 .....	120
Table 5.14: Logic Devices utilization by architecture 22 .....	120
Table 5.15: Logic Devices utilization by architecture 23 .....	120
Table 5.16: Performance indices of architecture 21.....	121
Table 5.17: Performance indices of architecture 22.....	121
Table 5.18: Performance indices of architecture 23.....	121
Table 5.19: the filtering results for greyscale ( $64 \times 64 \times 8$ ) MRI volume data of the generic 3-D FIR Sharpen operators using architecture 21 .....	122
Table 5.20: the sharpening results for k-space greyscale ( $64 \times 64 \times 8$ ) MRI volume data of the generic 3-D FIR Sharpen operators using architecture 22 .....	123
Table 5.21: the filtering results for greyscale ( $64 \times 64 \times 8$ ) MRI volume data of the generic 3-D FIR Sharpen operators using parallel architecture 23.....	124

## List of Equations

(2.1).....	27
(3.1).....	31
(3.2).....	31
(3.3).....	32
(3.4).....	32
(3.5).....	32
(3.6).....	33
(3.7).....	33
(3.8).....	33
(3.9).....	33
(3.10).....	34
(3.11).....	35
(3.12).....	35
(3.13).....	35
(3.14).....	35
(3.15).....	36
(3.16).....	36
(3.17).....	51
(3.18).....	51
(4.1).....	54
(4.2).....	54
(4.3).....	55
(4.4).....	59
(4.5).....	60
(4.6).....	60
(4.7).....	60
(4.8).....	61
(4.9).....	62
(4.10).....	62
(4.11).....	62
(4.12).....	63
(4.13).....	63
(4.14).....	72
(4.15).....	72
(4.16).....	72

(4.17).....	72
(4.18).....	81
(4.19).....	84
(4.20).....	84
(5.1).....	89
(5.2).....	89
(5.3).....	89
(5.4).....	94
(5.5).....	94
(5.6).....	95
(5.7).....	96
(5.8).....	96
(5.9).....	96
(5.10).....	97
(5.11).....	97
(5.12).....	108
(5.13).....	108
(5.14).....	108
(5.15).....	122
(5.16).....	125
(5.17).....	125



# Chapter 1. INTRODUCTION

## 1.1 Background

Flynn and Rudd [1] stated in 1996 that “the future is parallel”. Currently, there has been an increasing interest in the development and use of parallel hardware platforms, e.g. FPGA, as well as parallel algorithms in one and multi-dimensional processing applications such as weather forecasting, biomedical imaging and robotic vision requesting high speed and high throughput rates. One and multi-dimensional digital convolution and correlation operations are widely used for digital image processing applications such as image filtering, enhancement and recognition [2-14]. As the number of arithmetic operations is very large and the demand for real time high resolution images is ever increasing, this computation requirement becomes extensive and the need for high-performance parallel image processing algorithms is becoming more important [15].

Previously, the main hardware platform to realize high performance parallel architecture was the ASIC. However, with the advances in semiconductor technology the capacity and performance of FPGAs have improved to such an extent that, together with their inherent parallelism and reconfigurability, these devices have become a viable prototyping hardware platforms for the investigation and implementation of high performance processing algorithm [16, 17]. Subsequently, FPGAs are now used, extensively, in modern high-performance filtering applications such as medical imaging, portable image, mobile video applications, satellite data, weather forecasting and seismic data, to mention few in enormous parallel algorithms applications that are mainly implemented utilizing FPGA as a reconfigurable hardware platform.

However, as the complexity on FPGAs has increased, several inadequacies in the software support tools, associated with FPGA realizations are becoming apparent and need to be addressed [16-20]. First, there is a lack of suitable constructs in Hardware Design Language (HDL), used in early design phase, to efficiently describe the structures to implement the parallelism existing in multi-dimensional convolution/correlation algorithms and their parallel filtering architectures, being realized in FPGA technology. Second, to facilitate the rapid prototyping of architectures, an IP library of commonly used cores needs to be developed, which would also include an efficient mapping strategy for the fundamental blocks onto FPGA structure. Finally, as the complexity of the systems being implemented on an FPGA

device increased, there is a requirement for a concurrent hardware/software system design flow.

As the application area for parallel processing increases, together with the amount of data to be processed, there is a growing requirement for the development of highly efficient FPGA implementations for a family of parallel filtering algorithms, for one and multi-dimensional applications in FPGA. Some of the related works in this area of research are outlined in the following section.

## **1.2 Related Research Works**

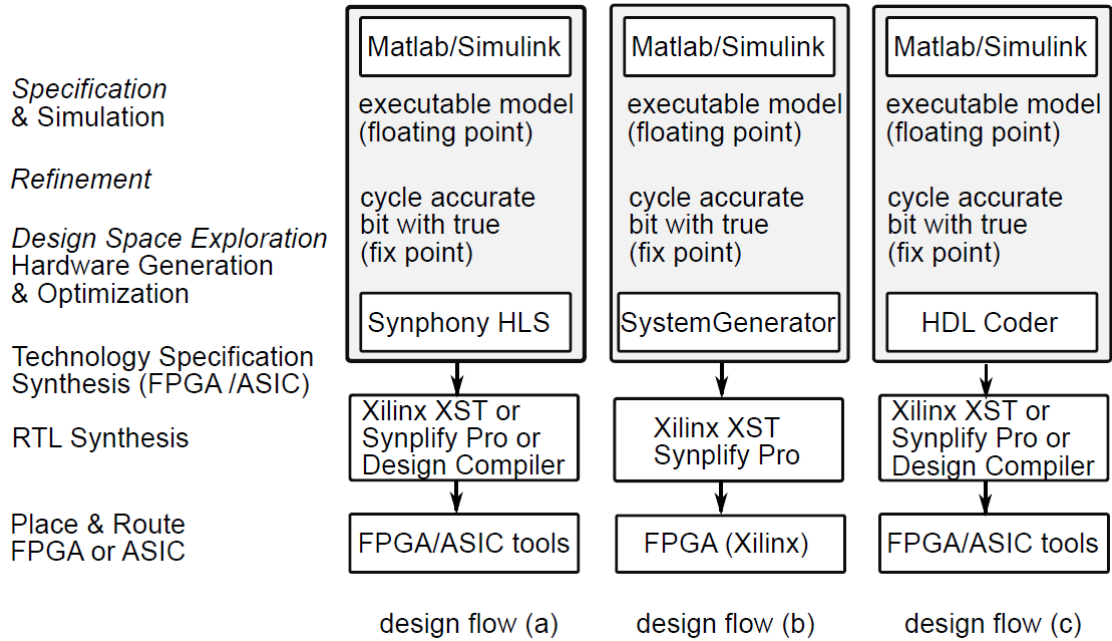
Although, there are many papers published on parallel processing and implementation, a few focus specifically on FPGA-based implement parallel filtering algorithms in 1-D, 2-D, 3-D and 4-D. Nevertheless, the research papers found in the literature can be classified into the following broad groups:

### ***1.2.1 Parallel Field Programming Tools***

FPGA implementation of parallel multidimensional filtering algorithms goes beyond the low-level line-by-line hardware description language programming [21]. High-level abstract hardware-oriented parallel programming tools are required to bridge this gap. Fortunately, the tools for implementing parallel filtering algorithms on FPGA enable parallel field programming at a higher-level abstraction. A comparison study was undertaken by Zoss et al [22] of three parallel design flow tools that adopt the modern specify-explore-refine concept, for the hardware realization of parallel Multi-dimensional convolution/correlation algorithms developed in the MATLAB / SIMULINK framework, as shown in Table 1.1.

The specify-explore-refine concept starts with the specification of an executable floating-point model, design exploration by fixed-point design followed by design space refinement with parameters influencing the architecture by resource allocation and a final scheduling. The three hardware generation tool chains are tightly coupled with the MATLAB/SIMULINK design and simulation tools. In the design flow of Table 1.1(b), the toolbox needed for the System Generator hardware generation is specific to Xilinx FPGAs. Both design flows in Table 1.1(a) and (c), however, have no restrictions with respect to FPGA or ASIC target technologies.

**Table 1.1: parallel signal processing algorithm implementations in FPGA/ASIC.**



Another comparison study [23] was undertaken into software design tools to implement parallel multi-dimensional convolution/correlation algorithms in FPGAs that accelerate the migration from traditional software algorithms to faster hardware implementations. In this study, MATLAB/SIMULINK is suitable for the simulation of parallel multi-dimensional filtering algorithms that are intended to be implemented on hardware, because of its timing simulation feature in the time and frequency domains with flexible presentation formats for easy viewing. This study compared Xilinx System Generator with Altera DSP Builder for automatically translating Simulink models into synthesizable hardware descriptions to be used with FPGA implementation tools of ISE and Quartus II development suite respectively. These tools provide Simulink libraries including common DSP, arithmetic, bus manipulation, control logic, storage, imaging, or communication functions. Advanced options such as HDL co-simulation and hardware-in-the-loop are also supported. HDL co-simulation allows designers to import legacy or new HDL code, and simulate it directly from the Simulink framework. Hardware-in-the-loop allows designers to verify designs in hardware directly from Simulink.

In a study [24] of visual data flow environments, the combination of tools and IP libraries helps the system designer manage design complexity, provides a flexible modelling framework, and facilitates migration from algorithms into silicon. In this study, the System Generator can maintain an abstract development level for modelling

and designing parallel multi-dimensional filtering algorithms that map designs into hardware implementations that are faithful, synthesizable, and efficient without substantially compromising the quality of either the functional representation or the performance of the hardware implementation.

### ***1.2.2 Parallel 1-D Signal Filtering Architectures***

Parallel 1-D raw signal filtering algorithms have been efficiently implemented as temporal or FFT convolution/correlation architectures using either DSPs or FPGAs.

In [25] DSPs were exploited. A one dimensional parallel block filter algorithm based on the overlap-add approach was implemented on multi-DSPs platform in the ASP-PI5 DSP card. An input of length ( $N = 4035$ ) and a variable length impulse response filter ( $m = 15, 31, 61$ ) were used; FIR filtering was carried out using the complex Fast Fourier transform (FFT) transform provided by the DSP library of functions. 1-D filtering results were obtained using single DSP processor and parallel 4-DSP system. The 4-parallel DSP system achieved a high speed up factor, close to the number of processors used. The performance indices of logic area, power consumption and throughput were not taking into consideration.

In the work undertaken by Hasan et al [26, 27], the performance indices of area, speed and power consumption are considered as a complete package. A generalized parallel 1-D signal filtering algorithm is implemented as a parameterized efficient FPGA-based architecture using Xilinx System Generator. The implemented algorithm is a linear spectral filter achieved by a parallel FFT/point-by-point complex inner product/IFFT convolution unit array. An input of real-time speech (22050 KHz /1Ch/16 bit) sequence was filtered using a distinctive FIR filter impulse response kernel of (3, 7, 15, 31, 61, 127, 255, 511, 1023 and 2047) coefficients to be individually applied with a FFT/IFFT N-point (8, 16, 32, 64, 128, 256, 512, 1024, 2048 and 4096) respectively. The parameterized implementation provides a rapid system-level FPGA prototyping and operating frequency flexibility. Consequently, the results are readily obtained for two targeted Virtex-6 FPGA boards, namely xc6vlX240T and xc6vlX130T, with a low total power consumption (static + dynamic) of 1.6 W and down to 0.99 W, respectively, at a maximum frequency of up to 216 MHz .

The work outlined in [26] has been cited in research undertaken in two other institutions [28, 29]. The first was the Microsystems Design Laboratory (MDL), Department of

Computer Science and Engineering, The Pennsylvania State University, where researchers developed efficient FPGA-based implementation for a reconfigurable Network-on-Chip platform [28]. The second paper will be mentioned in subsection 1.2.3.

The work discussed in [27] was cited by an application [30] of model based development of the digital part of a RFID transponder with Xilinx System Generator for Virtex-4 xc4vfx60 FPGA board. The reported approach proved the efficacy of usage of system-level abstraction of hardware-oriented programming, as an alternative to gate-level hardware descriptive language, to satisfy the conformant RFID product development at a minimal development-to-market time.

In the work carried out by Hwang and Ballagh [31] on the implementations of FIR filters using System Generator, the trade off between filter size and throughput was discussed by providing the performance results of three 64-tap FIR filters with a varying number of MAC engines. The performance indices of the occupied slices and speed were taken into consideration. The operating clock frequency was not particularly sensitive to the number of MAC-engines employed. A single-MAC architecture has the drawback that the throughput is inversely proportional to the number of filter taps. The throughput can be increased dramatically by exploiting parallelism that matches resource usage and availability to throughput, using System Generator.

### ***1.2.3 Parallel 2-D Image Filtering Architectures***

Parallel 2-D image filtering algorithms using spatial or spectral convolution/correlation architectures were efficiently implemented using either DSPs or FPGAs.

In [32], a 2D parallel block-filtering algorithm was implemented on the ASP-P15 card of 4-ADSP21060. The 2D algorithm was found to improve the performance of digital filtering systems by segmenting the 2D input into smaller block sizes, which led to a highly parallel implementation. The speed up results were presented for three different input image sizes (128×128), (256×256) and (512×512). Each input was filtered using fast number theoretic transforms. Only, the speed was considered out of the other performance indices of area, power consumption and throughput.

A research team led by Bouridane [33] successfully reported an interesting comparison for multi-processor platform and FPGA implementations of a parallel 2-D FFT single unit architecture using parameterizable structural of row-column processing. The FFT

algorithms were including radix-2, radix-4, split-radix and fast Hartley transform (FHT). These implementations were carried out under a common parallel architecture in order to enable the system designers to meet different system requirements.

Parallel 2-D FFT processor implementations were based on different FFT algorithms for matrix sizes  $N = 256$  and  $1024$ . Speed-up decreased with an increase in the number of PEs because of the increased delay due to memory conflict [33]. The performance results of parallel 2-D FFT implementation were subsequently displayed in terms of the maximum system frequency, chip area and throughput against the number of PEs for matrix sizes  $N = 256$  and  $1024$ . The chip area requirement and the throughput increased linearly as the number of PEs increased for all FFT algorithms, while the maximum system frequency slightly decreased.

The FPGA-based implementations of the parallel 2-D FFT algorithms, targeted on Virtex-2000E FPGA, presented a low-cost solution for 2-D FFT with real-time performance [33]. In addition, the FPGA implementation compared favourably in terms of area and area/speed ratio with multi-processor implementation. The throughput was stable at 35 FPS for  $512 \times 512$  image. The performance indices that were taken into consideration are occupied slices, speed, throughput and the power consumption that was inferred from the area occupation.

In [34] a parallel 2-D Image filtering algorithm was implemented on a Xilinx XC4V5X35 FPGA board using a spatiotemporal convolution implementation, suitable for portable image processing applications. The system reduces the power consumption while still maintaining video rate operation. This paper describes how splitting the data stream into multiple processing pipelines can reduce power consumption in contrast to the traditional spatial (pipeline) parallel processing technique. Real-time image processing system functions ( $3 \times 3$  Sobel filters and  $3 \times 3$  anisotropic diffusion) were implemented to show the principle of the technique. A relation was observed between the frame processing time achieved and the consumed power when processing the images without any partitioning, partitioning into three and 6 partitions respectively. The effect of this image partitioning technique was strongest when faster processing times were required. Particularly, when real time image smoothing were using the anisotropic diffusion algorithm, which required large number of iterations to process one frame. For example, if 100 iterations were required and each iterative sub frame processed at 0.6 ms, a total frame time will takes 60 ms with a final video rate of 15Hz,

running one partition at 300 MHz, or three partitions at 100MHz. The resultant difference in power consumption is 261mW. In the case of smaller iterations the background power consumption of the FPGA dominates. The performance indices gave increasing benefits with shorter processing times, and up to 45% drop in the power consumption. The performance indices did not take into considerations the throughput.

In [21, 35], nine parallel 2-D grayscale MRI spatial convolution algorithms were developed as one generic architecture, targeted on two Virtex-6 FPGA boards, namely, xc6vlX240T and xc6vlX130T. This generic architecture was used as “ bulge and develop” processor to improve the nine MRI image filtering operators for generating enhanced MRI scans filtering results without significantly affecting the developed performance indices of high throughput and low power consumption at maximum operating frequency.

This approach of development, improvement and implementation was accomplished using Xilinx System Generator. Where, a single generic architecture was efficiently prototyped to achieve high filtering performance of (11230 FPS) throughput for 64×64 MRI grayscale scan, minimum total power consumption of 0.86 Watt with a junction temperature of 52°C and a maximum frequency of up to (230 MHz). The improved generic architecture provides visibility enhancement within the filtered MRI scan to aid the physician in detecting brain diseases, e.g., trauma or intracranial haemorrhage. The high filtering throughput is feasibly nominee the nine parallel MRI filtering algorithms for applications such as real-time MRI potential future applications.

The research work described in [21] was cited in four interesting and related research papers in parallel 2-D image filtering algorithms which are FPGA- based implementation using Xilinx system generator. The first paper [29] was gesture recognition application using field programmable gate arrays, the performance indices of its architecture was only the logic area occupation. The second paper [36] was teaching and research program in FPGA based Digital Signal Processing using Xilinx System Generator, the implementation of various designs was carried out on a Xilinx Spartan-3E FPGA, no performance indices were mentioned. The third paper [37] was a VLSI Implementation of an Edge detection system for images, its implementation was developed in Spartan 3A DSP FPGA, the performance relates only to the logic area occupation. The fourth paper [38] was an efficient FPGA implementation of MRI image filtering and tumour characterization using Xilinx System Generator, six (3×3) edge

detection, Gaussian blur, thresholding & edge sharpening algorithms were implemented on a Spartan 3E starter kit (XC3S500E-FG320), the performance indices considered were only the logic area occupation and clock speed of 50 MHz .

#### ***1.2.4 Parallel 3-D Volumetric Data Image Filtering Architectures***

Parallel 3-D raw volume image filtering algorithms were efficiently implemented as 3-D spatial or 3-D spectral convolutions /correlations architectures using either multi-DSPs platform or FPGAs boards.

In the research papers [39, 40], a parallel 3-D spatial convolution algorithm was implemented on a multiprocessor DSP system (the ASP-P15 Quad-DSP card), consisting of 4-SHARC DSP processors (ADSP21060), and benchmarked against a single SHARC-DSP processor system, to obtain the performance index of more than 5-fold speed up factor. This method is suitable for high resolution / high speed 3-D image and video processing. The proposed 3-D parallel filtering algorithm eliminates the overlapping segments overhead in the block-filtering method, and the boundary conditions in parallel filtering applications. When the system impulse response is large, the overall memory distribution of the parallel system was enhanced by segmenting both the 3-D input data and the impulse response of the system into smaller independent subsections that can be simultaneously processed. The input 3-D volume image was of size (64×64×64).

In the research papers [41, 42], a parallel 3-D fast transformed convolution algorithm was implemented using a 3-D New Mersenne Number Transform (3-D NMNT) and 3-D vector radix fast Hartley transform (3-D VR FHT), with input 3-D volume images of size (32×32×32) and (64×64×64) respectively. The implementations were realized in a multiprocessor DSP system (the ASP-P15 Quad-DSP card), consisting of 4-SHARC DSP processors (ADSP21060), and benchmarked against a single SHARC-DSP processor system, to obtain the performance index of more than 16-fold speeding factor. The same parallel 3-D filtering algorithm structure exploited in [39, 40] was implemented , but, in fast transformed convolution.

In [43], a parallel three-dimensional 3-D FIR digital filter algorithms were designed by decomposing the 3-D complex design specification into one-dimensional (1-D) complex design specifications, that can be regarded as frequency response specifications of 1-D FIR filters. Then a set of 1-D FIR filters were designed to approximate them. Finally,



combining the resulting 1-D filters gave a linear phase 3-D filter with parallel structure. The technique can create linear phase 3-D filters by merely designing 1-D FIR filters. Furthermore, the structure of the resulting 3-D filter will have high degree of parallelism, modularity and regularity, and is suitable for VLSI implementation and parallel signal processing. No performance indices were presented.

In an article undertaken by Lin et al [44], the research and development of a massively parallel three-dimensional (3-D) spatial and FFT convolution architecture was described with its programming technology, in construction of parallel video processing components, and in development of video processing applications. The implementation was carried out using multi-core architectures for video processing. An archetypal example of optimizing the parallelism of a video processing application was considered. The 3-D convolution algorithm was implemented as either 3-D spatial convolution or the 3-D FFT convolution, where, the 3-D volume video input is  $\mathbf{V}$  having  $n_V$  number of pixels, and the 3-D FIR filter kernel is  $\mathbf{K}$  having  $n_K$  number of pixels. Then a spatial convolution method required  $O(n_V n_K)$  operations, while the Fast Fourier transform (FFT) convolution perform the 3-D convolution in  $O(n_V \log_2 n_V)$  steps. Since,  $n_K$  is always greater than  $\log_2 n_V$  for long 3-D kernel, then, the Fourier multiplication technique should quickly win over spatial multiplication for long FIR kernels. For short 3-D kernel, however, the inequality formula will be  $n_K < \log_2 n_V$ , hence, the implementation should be carried out using the 3-D spatial convolution.

In [45] , a massively parallel Correlation Census algorithm was implemented on Altera Stratix 1S40 FPGA with real time performances and an output rate up to one hundred images per second. The main objective of this algorithm was to search for the correspondence between two input images (right and left) taken from two different angles to get a "depth map" shape to reconstruct the 3D scene. To compute this algorithm, many parallel-pipelined pre-processing stages are needed (means calculus, windowed Census transformation, best correspondence searching, and image filtering). This real time parallel architecture was developed for the PICASSO project to achieve the calculus of depth map to get (100 FPS) which is needed in many applications such as medical surgery, robotics, vehicle driving assistance and many other applications.

### **1.3 Research Aims and Objectives**

One and multi dimensional raw data collections introduce noise and artifact need to be recovered from degradations by an automated filtering system before further machine

analysis. The need for automating wide-ranged filtering applications necessitates the design of generic filtering architectures. This would enable the development of multidimensional and extensive convolution operators. Consequently, the aim of this thesis is to investigate the problem of automated construction of a generic parallel filtering system. To achieve this goal, performance-efficient FPGA implementation architectures are developed to realize parallel one/multi-dimensional filtering algorithms. The proposed generic architectures provide methods of fast FPGA prototyping for high performance computations to obtain efficiently implemented performance indices of area, speed, dynamic power, throughput and computation rate, as a complete package. These parallel filtering algorithms and their automated generic architectures tackle the major bottlenecks and limitations of existing multiprocessor systems in terms of wordlength, input data segmentation, boundary conditions as well as inter-processor communications, in order to support high data throughput real-time applications of low-power architectures using a Xilinx Virtex-6 FPGA board.

### ***1.3.1 Aims***

The aims of this PhD research project are to introduce highly efficient FPGA implementations of parallel multi-dimensional data convolution/correlation algorithms for the digital filtering applications in 1-D, 2-D and 3-D up to 4-D to provide generic architectures with minimum logic area, low dynamic power consumption, fast speed and high throughput. The development tool of System Generator within the Xilinx ISE design suite are exploited to realize these implementations. VIRTEX-6 family of the 40 nm FPGA is the hardware platform for the novel instantiation of the parallel multi-dimensional filtering algorithms.

As the FPGA is to be the implementation platform from which a range of performance parameters for various algorithms will be obtained and compared to other implementations. It is essential to highlight various architectural features of FPGAs which will give potential implementation advantages. For example, the inherent parallelism within the FPGA needs to be analyzed together with its flexible wordlength compared to the DSPs sequential processing interprocessor communication and fixed wordlength bottlenecks. Moreover, parallel field programming languages are studied to exploit the most efficient performance indices. Therefore, parallel multi-dimensional data filtering algorithms are studied and adapted to be mapped into parallel generic architectures.

### ***1.3.2 Objectives***

The objectives of this research project are based on the achievements of the above-mentioned research aims. Therefore, in Chapter 3, parallel block-filtering architectures are studied and developed in 1-D, appropriate for the parallel 1-D raw signal filtering algorithm implementations in the Virtex-6 FPGA board. The developed parallel 1-D filtering architectures are performance-efficient in the temporal and frequency domain to cover the short and long 1-D FIR kernel real-time applications.

A new and novel parallel multidimensional image filtering architectures are introduced, in chapter 4 and 5, and their implementations as a generic architectures are described. The parallel 2-D, 3-D and 4-D image filtering algorithms are suitable for the calculation of the multidimensional data convolutions/correlations, and their related applications. These parallel filtering architectures solve the problems associated with the parallel block-filtering algorithms of boundary conditions and overlapping segments, as well as the parallel processing stages communication bottleneck. Furthermore, the developed parallel architectures are performance-efficient in the spatial and frequency domain to cover the short and long multidimensional FIR kernel applications.

## **1.4 Contributions per Chapter**

The primary contributions of this research project are:

### ***1.4.1 Contributions of Chapter 3***

In chapter 3, implementations of the parallel 1-D filtering algorithms are presented and mathematically modelled using the 1-D block filtering method. The architectures are implemented on FPGA, as parallel temporal and FFT architectures with attractive performance indices. Five generic architectures were implemented on Virtex-6 ML605 board. Also, a complete set of area, speed, power, throughput and computation rate for each architecture were obtained. An example of a practical application of the above algorithm is the realization of a cross-correlation function using a parallel 1-D matched filter algorithm for real-time speech signature detection. Subsequently, this application was mathematically modelled, implemented and analyzed.

### ***1.4.2 Contributions of Chapter 4***

In chapter 4, parallel 2-D grayscale/colour image filtering algorithms are discussed, mathematically modelled and implemented using 2-D input decimation and output image reconstruction by interpolation. The FPGA implementations, as parallel spatiotemporal and FFT architectures, were realized. Ten FPGA-based architectures

were implemented on Virtex-6 ML605 board. Also, a complete performance indices set were obtained for each of the 2-D filtering architectures. A 2-D image generic processor were also developed using mutli-MAC FIR units. A 3-D Colour image processor were devised to act as an open development 3-D colour filtering engine. A parallel Cross-correlator engine was also developed as a parallel 2-D matched filter algorithm to locate any MRI slice within a MRI data stack library. Twelve 3-D MRI filtering operators, edge detection and noise-smoothing, were also developed to improve biomedical imaging results.

### ***1.4.3 Contributions of Chapter 5***

In chapter 5, a novel parallel 3-D grayscale/colour volumetric image-filtering algorithm is presented using volume data image segmentation by decimation and output reconstruction by interpolation. Eight architectures were developed and implemented on Virtex-6 board. Fourteen 3-D MRI filtering operators were also developed and used to improve biomedical imaging results. As a practical example of the above architectures, a four dimensional coloured MRI (fMRI) filtering processor was realized using 3-D spatial architectures. Two examples were realised using the 3-D FFT architectures to filter k-space MRI volume data and 3-D cross-correlator.

## **1.5 Organization of the Thesis**

The content of the remaining chapters are outlined below:

**Chapter 2** introduces and illustrates the parallelism concept within parallel multi-dimensional data filtering algorithms and the realization of these parallelism types on the FPGA. Therefore, the use of the Xilinx System Generator in the FPGA implementation of the multi-dimensional filtering algorithms is discussed, followed by a description of the performance related factors is when comparing various implementations of parallel algorithms. The chapter also outlines the seven key steps involved in the implementation of such parallel filtering algorithms.

**Chapter 3** presents the adaptation of the generalized parallel 1-D speech signal filtering algorithms based on the block filtering method. The algorithm layout and mathematical model for the general filtering case are given. Furthermore, the implementation of five generic architectures on Virtex-6 board are discussed together with their performance indices. The chapter ends with a description of the implementation of a real-time speech signature detection using parallel 1-D cross-correlation architecture.

**Chapter 4** presents a new generalized parallel gray/colour image filtering algorithms based on the input decimation and output interpolation. The algorithm layout and mathematical model for the general filtering case are given. The implementation of ten generic architectures in Virtex-6 ML605 board is discussed together with their performance indices. These generic architectures can be applied as a colour image processors. The chapter ends with a description of the implementation of a 2-D cross-correlation application to target an MRI slice within a library of MRI stack.

**Chapter 5** introduces a novel generalized parallel gray/colour volumetric image filtering algorithms based on the input decimation and output interpolation. The algorithm layout and mathematical model for the general filtering case are given. The implementation of eight FPGA-based architectures in Virtex-6 ML605 board is described as well as their performance indices. Three successful applications are developed.

**Chapter 6** presents summary of the original contributions, overall conclusions and investigates potential research projects for future work in the parallel multi-dimensional data filtering algorithms.

## Chapter 2. **Parallelism Analysis in Filtering Algorithms and FPGA-Based Architectures**

### **2.1 Introduction**

An FPGA is a general-purpose integrated circuit that is field “reprogrammed” even after it has been deployed into an embedded system. FPGAs have increasingly become to be seen as complete systems because of their increased capacity, improved performance, inherent parallelism and reconfigurability [16, 17, 46-49]. Consequently, FPGAs are being exploited extensively for the hardware realization of modern parallel one and multidimensional data filtering algorithm applications in 1-D [26, 27, 33, 50-54], 2-D [3, 21, 33, 55-61], 3-D [45, 62-68] and 4-D [69-73] such as real-time audio and speech filtering, filtering medical imaging, portable image filtering, mobile filtering applications, robot vision, satellite data filtering, weather forecasting, seismic data filtering and wireless communication system. To mention few in enormous parallel algorithms applications that are mainly implemented utilizing FPGA as a reconfigurable hardware platform.

FPGA is programmed by downloading a configuration program called a bitstream into static on-chip random-access memory. Similar to the object code for a microprocessor, this bitstream is the product of compilation tools that translate the high-level abstractions into low-level and executable equivalent code. Xilinx System Generator can compile an FPGA program from a high-level Simulink model.

FPGA provides an array of configurable resources that can implement a wide range of arithmetic and logic functions. These resources include dedicated DSP blocks, multipliers, dual port memories, lookup tables (LUTs), registers, tri-state buffers, multiplexers, and digital clock managers. In addition, Xilinx FPGAs contain sophisticated I/O mechanisms that can handle a wide range of bandwidth and voltage requirements. The DSP computing and I/O resources are linked under the control of the bitstream by a programmable interconnect architecture that are wired together into systems.

FPGAs are high performance data processing devices. Digital signal processing performance is derived from the FPGA’s ability to construct highly parallel architectures for processing data. In contrast with a microprocessor or DSP processor, where performance is tied to the clock rate at which the processor can run, FPGA performance is tied to the amount of parallelism that can be structured in the multi-

dimensional filtering algorithms. A combination of increasingly high system clock rates and a highly-distributed memory architecture empowered the ability to exploit parallelism in filtering applications that operate on data streams. For example, the raw memory bandwidth of a large FPGA running at a clock rate of 600 MHz can be hundreds of terabytes per second.

When working in System Generator, it is important to keep in mind that an FPGA has many degrees of freedom in implementing Parallel multi-dimensional filtering algorithms. The freedom also exists to define data path widths throughout the architecture, and to employ many individual data processors (e.g., MAC engines and FFT) with high abstractions.

The remainder of this chapter is organised as follows:

In section 2.2, three different ways in which parallelism, namely temporal, spatial and logical, is used in parallel filtering algorithms, are discussed. In section 2.3, the main characteristics of FPGA-based implementation of parallel filtering algorithms are presented. In section 2.4, the FPGA implementation of parallel digital filtering is described. In section 2.5, overall FPGA implementation steps are explained with a flow chart. In section 2.6, parallel field programming using XSG tool are discussed. In section 2.7, performance factors are described as a complete package when comparing various algorithm implementations. Finally, the conclusion of this chapter is presented in section 2.8.

## **2.2 Parallel Filtering Algorithms**

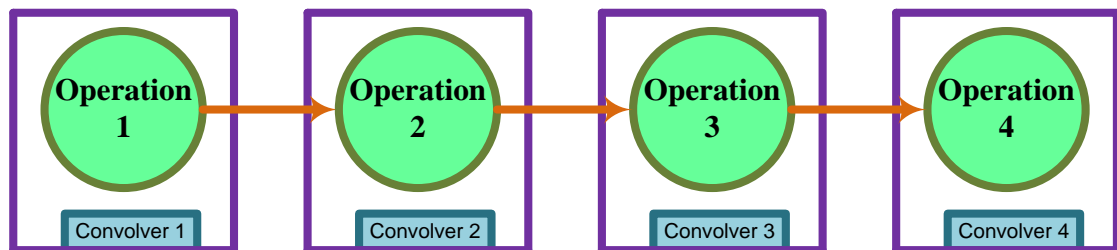
Parallel one and multi-dimensional filtering algorithms are concurrent computation models which tend to be used in applications where there is a demand to process lots of information with the sampling rates that can range from kHz as in speech environments, right through to MHz, in the case of image processing applications [16]. The design of these parallel filtering algorithms to solve a particular problem is strongly influenced by the hardware platform and the software development tools.

Factors that affect the performance of such algorithms on a particular hardware platform are dependent on the degree of parallelism and the overhead incurred in scheduling and synchronizing the parallel tasks. Parallelism can either intrinsically exist in the filtering algorithms or can be introduced by organizing the computation to allow a parallel

implementation. Three parallelism types are observed in these filtering algorithms as temporal, spatial and logical parallelisms [17].

### 2.2.1 Temporal Parallelism

The structure of parallel filtering algorithm suggests a separate convolver for each filtering operation, as shown in Figure 2.1. This is a pipelined structure in which the data are processed in passing through each stage. Each convolver operates and passes the result to the next stage. The total filtering time will not be reduced if each successive convolver has to wait until the previous convolver completes processing. Nevertheless, the throughput can be increased if each convolver operates on an independent part of the data.



**Figure 2.1: A temporal parallelism is exploiting an independent convolver and networking using pipeline**

Pipelining an operation can significantly improve the performance of the parallel filtering algorithm when a downstream convolver may begin operating before the upstream convolver due to the low latency of an operation. Operation latency is the time required between the data input and the corresponding output is available. In a hardware platform, the total throughput is given by the sum of each convolution stage latency.

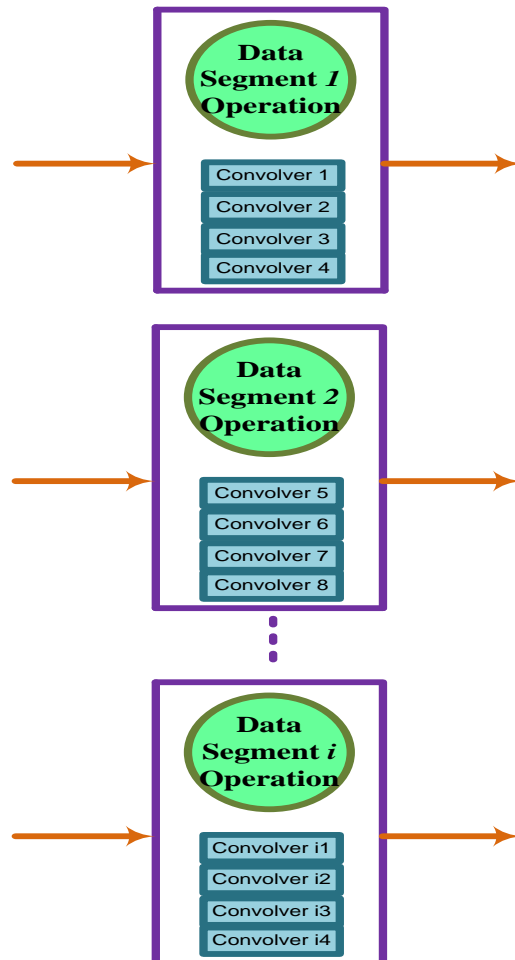
### 2.2.2 Spatial Parallelism

This type of parallelism, as shown in Figure 2.2, can occur internally within the operations level of the parallel algorithm, when many iterative operations are performed on a certain data segment. This internal parallelism may be exploited by partitioning the data into segments and using separate convolvers to perform the operation on each segment.

For example in video filtering, the image sequence may be partitioned in time by assigning successive frames to separate convolvers. The spatial parallelism considerably



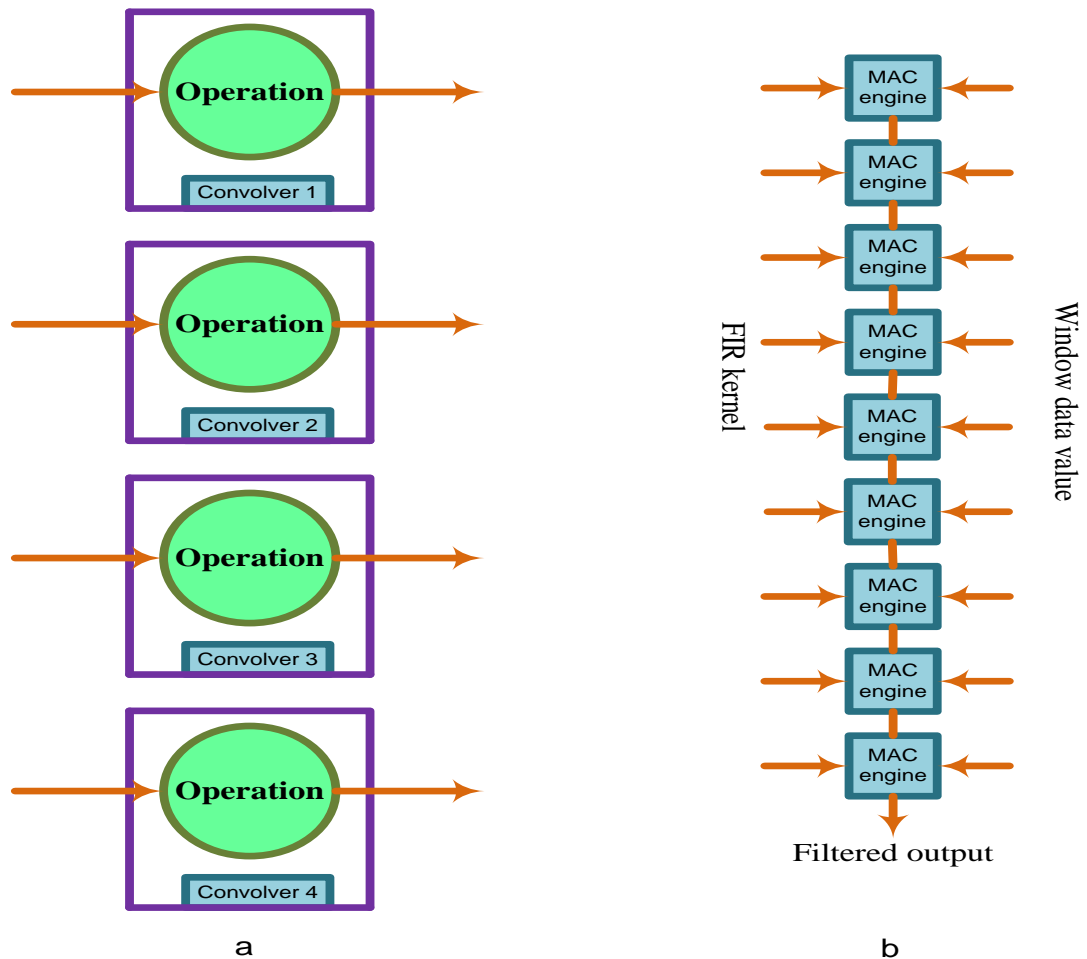
minimise or even eliminate the communication among convolvers, due to the independent data segments, hence, no accessing of shared resources. Consequently, each convolver must have some local memory to reduce any delays associated with contention for global memory.



**Figure 2.2: Spatial parallelism is dedicating parallel convolvers for each data segment**

### 2.2.3 Logical Parallelism

The third form of parallelism which may be exploited in a regular structure where an operation or the functional block can be reused and arranged in parallel is illustrated in Figure 2.3, (a). A linear convolution algorithm multiplies the data value within a window by a coefficients set or kernel. These multiply and accumulate (MAC) blocks are repeated many times. This is a classic example of logical parallelism, as illustrated in Figure 2.3, (b).



**Figure 2.3: Logical parallelism exploits an independent Convolver for each input segment as in (a) and a classical example is in (b)**

### 2.3 FPGA-Based Implementation of Parallel Filtering Algorithms

An FPGA can readily implement most of the parallelism types [17]. Since FPGAs are inherently parallel, then any logic area required by the parallel multi-dimensional convolution algorithms can be implemented on FPGA by reconfiguring separate hardware architectures for each convolution operation. This permits the FPGA to be a reconfigurable device of choice for rapid prototyping and development of such parallel filtering algorithms. Moreover, the FPGA has the speed that results from a hardware design while retaining the reprogrammable flexibility of software tool.

Temporal parallelism can be implemented, as a pipelined architecture, by reconfiguring a separate convolver for each convolution operation in the pipeline. In this data synchronous algorithm, the data is passing from the output of one operation to the input of the next. If the data cannot be synchronized, then appropriate buffers may be incorporated to manage the variation in the dataflow or access patterns. Spatial parallelism may be exploited by reconfiguring multiply copies of the convolution hardware and assigning independent input data partitions to each of the copies. These

map readily onto the available resources of the FPGA. Logical parallelism is well suited to FPGA implementation, and may be accelerated significantly. This accomplished by the parallel hardware and unrolling sequentially performed operations.

The partitioned input data are serially streamed into the parallel multi-dimensional filtering stage. This can be realized as the FPGA implementation, especially when interfacing directly to a camera or vision system. If all the convolutions/correlations can be implemented using a streaming processing, then the implementation of the entire filtering algorithm as a single streamed pipeline results in a very efficient on-the-fly filtering architecture. The required throughput can usually be achieved by pipelining.

The ability to significantly exploit the inherent parallelism of the FPGA has considerable implications when reconfiguring an embedded architecture for the parallel filtering algorithms. Performing multiple serial convolution in parallel enables the clock speed to be lowered noticeably. A streamed pipelined architecture implemented on an FPGA can often operated at the native input or output clock frequency. This corresponds to a reduction in the clock speed of a serial convolver of two magnitude orders or more. The dynamic power consumption of this parallel architecture is directly related to the clock frequency, thus an observable lower power design is obtained.

The development of an FPGA-based design for the entire filtering algorithm will allow the embedded realization within only one or two chips, depends on FPGA type. Consequently, enabling a complete parallel multi-dimensional filtering algorithm to be embedded within a sensor or camera. Thus, FPGA-developed architectures can be then be embedded within many portable applications.

#### **2.4 Parallel Digital Filtering using FPGAs**

Before FPGAs, DSP circuits, e.g. DSP and Gate arrays, could be constructed, but with less flexibility to run parallel algorithms in software due to fixed wordlengths, e.g. 16 bits, 24 bits, 32 bits. Operations with shorter wordlengths still required one execution of the entire wordlength. On the other hand, FPGAs allow the choice exactly of the required wordlength. FPGAs enable configuration of data paths into arbitrary sizes, allowing a trade-off between resolution (precision) and parallelism. An additional benefit of minimizing precision comes from shorter propagation delays through narrower arithmetic units [68].

Note that the general speed-area complexity of an  $N \times N$  bit multiply is one quarter of the general speed-area complexity of a  $2N \times 2N$  multiplier. Therefore, if using a 16 bit processor to solve an 8 bit problem the arithmetic processing silicon is being used at 25% MAC efficiency. If using a 24 bit processor for an 8 bit problem then the processor is being operated at around 11% (1/9th) MAC efficiency. On the other hand, the FPGA wordlength is not constrained to the traditional 8, 16 or 32, but can be sized to the required arithmetic resolution of 5, 9, 13 or 21. Moreover, the FPGA-based implementations exploit the flexibility to change the resolution at different parts of an FPGA-based architecture.

Early gate-arrays were simply arrays of NAND gates. Combinational and sequential functions can be implemented by interconnecting the NAND gates. Early gate array implementation flow would be designed, simulated/verified using simulators and netlisters. Early GAs were usually used to implement some two level logic functions such as flip-flops, registers, addition and subtraction functions. Once a layer(s) of metal have been laid on a device, the GA cannot be changed, updated or fixed.

Therefore, digital signal processing moved to FPGAs, so that, the logic specified is changeable by “field” reprogrammed interconnection. FPGAs are carefully balanced repository of multi-input logic, flip-flops, multiplexers, memory, arithmetic resources and DSP components.

The modern concept of an FPGA implementation is to interconnect the available logic area to implement the algorithm using modern toolsets and design flows. However, new issues have emerged:

- Correct processing of arithmetic operations, i.e. overflow, underflow, saturate.
- Wordlength specification and the choice between rounding and truncation.
- Latency or delays used to maintain the synchronization.
- What clock rate to be used?
- Does the device place and route?
- What logic devices should be utilized?
- To compromise between logic fabric and IPs core utilization.
- Control over implementation parallelism.

Therefore, FPGA-based implementations flexibility is accomplished in terms of binary arithmetic using finite wordlengths for the required numerical precision. On the other hand, an efficient mapping of a parallel multi-dimensional filtering algorithm to FPGA

hardware can be achieved by leveraging our knowledge of DSP theory and implementation to finite wordlength effects within the targeted FPGA logic area.

## **2.5 Overall FPGA Implementation Steps**

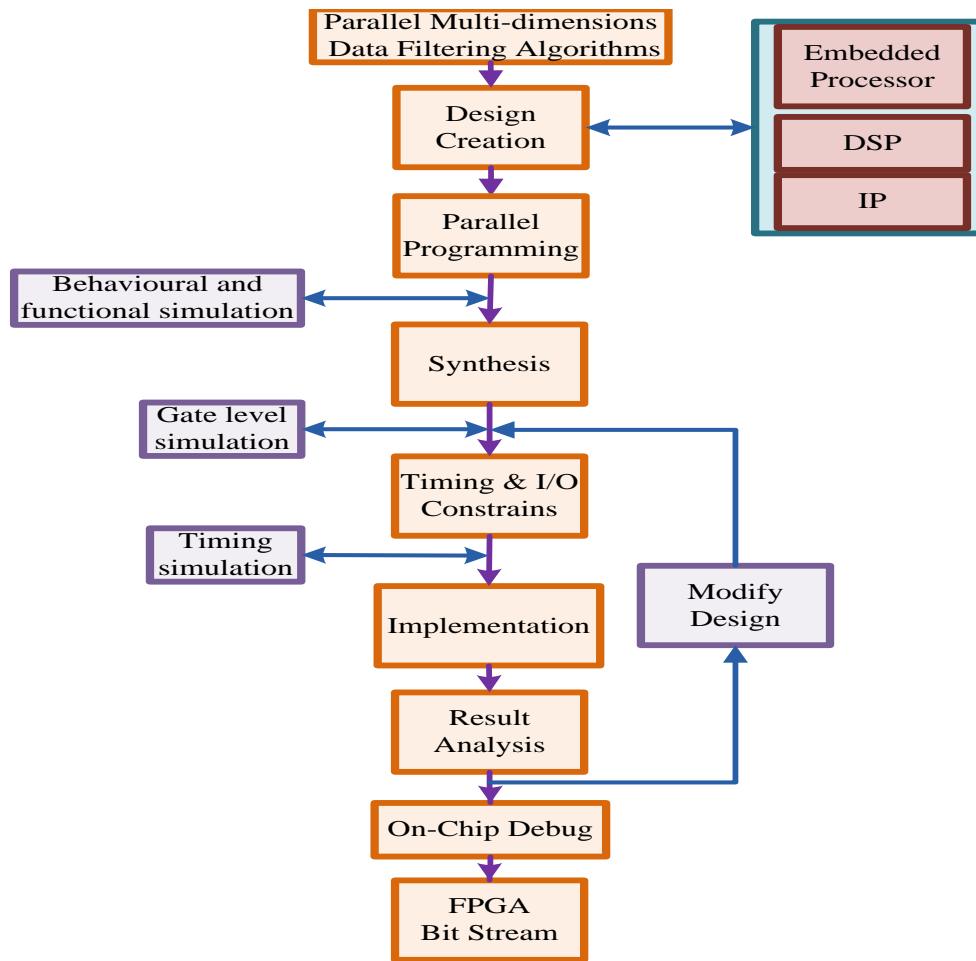
Generally, the overall FPGA implementation process of any parallel algorithm [22, 23] can be depicted as shown in Figure 2.4. This overall Xilinx ISE design flow is essential to FPGA implementation. Parallel algorithm implementation requires several key steps: Firstly, creating or adding parallel algorithm design sources in Verilog/VHDL files, schematic RTL entry, C/C++ embedded system files, IP core generator files or DSP System Generator circuits [74].

Secondly, Synthesis is the process of transforming the design sources into architectural specific gate level net-lists according to control aspects constraints. The interconnectivity is also defined between the building blocks. For interchange between different tools, a net-list is often represented using EDIF (electronic design interchange format). Thirdly, Timing and I/O constraints; the timing specifications in its basic forms define the clock operating frequencies as well as the timing requirements for all the main inputs and outputs. Fourthly, a parallel algorithm architecture implementation; the implementation process compiles the parallel algorithm architecture through individual phases of net-list translation, mapping and place and route. Initially, the net-list is mapped onto the target FPGA. In mapping, the logic is partitioned, merged or split to be fit into the available LUTs on that FPGA. The place and route phase joins together these mapped components and determines the routing required to connect the logic blocks memories and I/O.

Fifthly, the result analysis to verify that the parallel algorithm mapping objectives are accomplished. In addition, identify the implementation area that needs to be modified to reach an implementation closure. There are two main types of changes to improve the parallel algorithm implementation: either alter architecture properties of design strategies by modify the design source code or refine design constrains.

Sixthly, Hardware Debug is the most time consuming stage of the implementation cycle. Accordingly, the on-chip debugging must be highly detailed, at-speed internal FPGA signal capture and highly accurate viewing of post-place and route parallel algorithm implementation.

Seventhly, FPGA bit stream is the generation of a programming file that configures the FPGA implementation of the parallel algorithm via a parallel cable called JTAG.



**Figure 2.4: Overall FPGA implementation flow for Parallel Multi-dimensional Data filtering Algorithm**

In the sequel, the overall FPGA implementation flow provides a realization methodology in behavioural software and FPGA hardware for the parallel filtering algorithm. However, the FPGA implementation closure requires many design modification and iteration. In addition, the functionality and performance of the FPGA implementation can be verified via simulation at various implementation flow steps.

## 2.6 Parallel Field Programming using Xilinx System Generator Tool

Currently, FPGA goes beyond the low-level line-by-line HDL programming in implementing parallel multidimensional filtering algorithms. High-level hardware-oriented parallel programming methods can structurally bridge this gap [21].

At a higher level of abstraction, both parallelism and pipelining are well represented using a dataflow graph [17]. The FPGA parallel fabric outcomes in a better fit with visual dataflow. Indeed, most of the parallel filtering algorithms, such as signal and

image filtering, are often represented graphically by block diagrams. These blocks represent basic operations of addition and multiplication as well as complicated operations of convolution/correlation and FFT on the streamed data, and connection between blocks indicating the flow of data. Dataflow exposes the parallelism within multi-dimensional filtering algorithms without indicating the execution order.

This led to the logical approach of flied programming of the FPGA using dataflow languages, such as XSG [22-24, 74, 75]. System Generator is a system-level modelling tool that facilitates FPGA hardware design. It extends Simulink in many ways to provide a modelling environment that is well suited to hardware design. The tool provides high-level abstractions that are automatically compiled into the FPGA. The tool also provides access to underlying FPGA resources through low-level abstractions, allowing the construction of highly efficient FPGA designs. The design flow, as shown in Figure 2.4, involves developing the parallel algorithm within MATLAB, and then representing the algorithm graphically within Simulink using Xilinx library-based basic blocks.

The MATLAB/Simulink environment allows signals to be inspected using Scope and Display Simulink blocks, and written into the MATLAB workspace for plotting or further analysis, using the “To WorkSpace block”. In all cases, signals must be first output from the System Generator design via Gateway Out blocks. The System Generator also features a logic analyser as a WaveScope block, where signals from within the design can be probed. A further option is hardware testing with the ChipScope analyser. Using this technique, a special ChipScope block is added to the System Generator design, and signals of interest are connected to it. The whole design is then synthesised and implemented, and downloaded to the FPGA device. While the design operates, the ChipScope block captures signal data from the design running on the board, and these are transmitted back to the host PC, where signals can be viewed or analysed.

The Xilinx block set, as shown in Figure 2.5 , is used to implement the parallel algorithm for targeting Virtex-6 XC6VLX240T FPGA board. Xilinx Blocks set library consists of the following basic groups of blocks:

- Basic blocks for Multiplexer, Delay ...
- Communication blocks for forward error correction and modulator blocks ...
- Control logic blocks for control circuitry and state machines.

- DSP blocks of FFT, CORDIC blocks, Convolutional Encoder...
- Data type blocks that convert data types, includes gateway in and out.
- Math Blocks that implement mathematical functions of addition, subtraction ...
- Memory blocks that implement and access RAM, FIFO, shared memory...
- Utility tools blocks, e.g., System Generator token, resource estimation, HDL co-simulation, Black block and M-code block that provides interfaces to other software tools (e.g., ModelSim, FDATATool)

All the System Generator blocks are parameterised to allow the specification of data mathematical representation, type, wordlength, width and the IP core selection ... etc. The System Generator block sets are specifically targeted the basic and advanced DSP features within the Xilinx FPGA boards, resulting in an efficient development, implementation and use of resources. System Generator blocks are bit-accurate and cycle-accurate. Bit-accurate blocks produce values in Simulink that match corresponding values produced in hardware; cycle-accurate blocks produce sampling values at corresponding times.

The Xilinx blocksets map Simulink system parameters into entities and architectures, ports, signals and attributes in a hardware model. These mapped parameters are then converted into a hierarchical HDL netlist as well as the necessary command files to create an IP block netlist, which creates project and script files for HDL simulation, synthesis, placement, routing and bit stream generation.

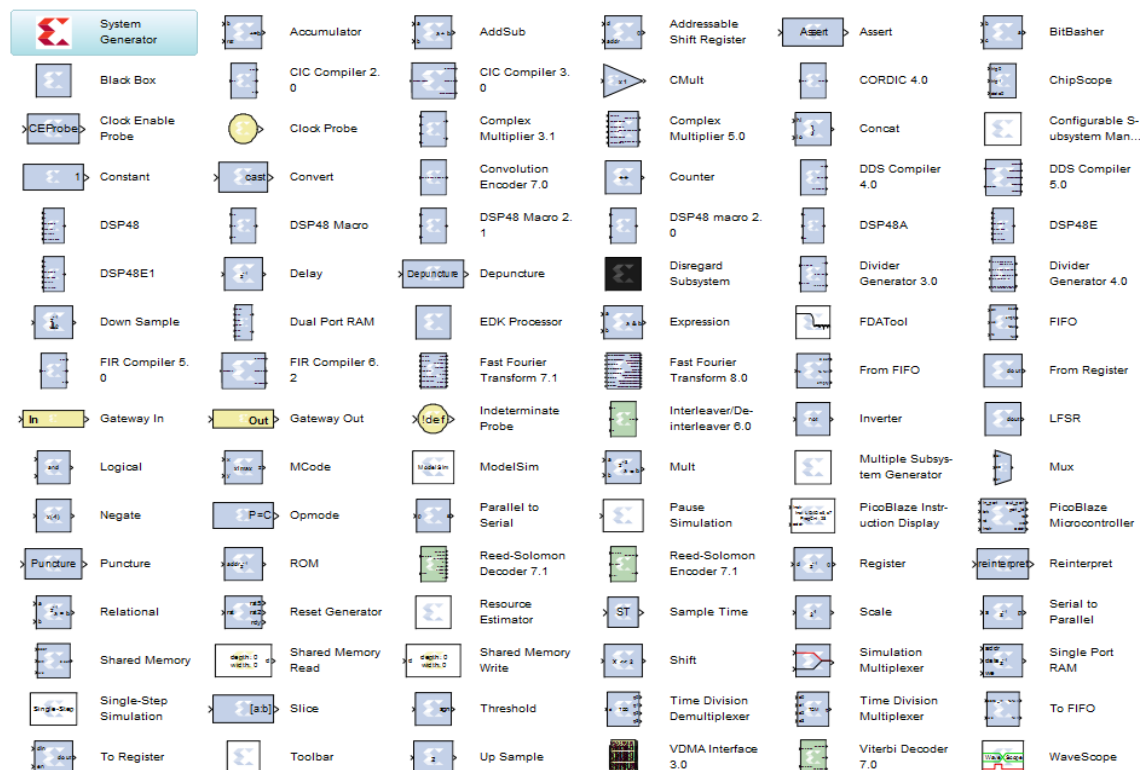


Figure 2.5: Xilinx System Generator tool complete block sets.



## **2.7 Performance Indices**

The performance indices of the parallel filtering architectures are considered as a complete set of area, speed, power, throughput and computation rate values using XSG to target a Virtex-6 ML605 FPGA board [76]. The area represents the occupied logic assets of FFs, LUTs, Slices, DSP and BRAM. The speed relates to the maximum clock frequency in (MHz). The power consumption is the dynamic power in (mWatt). The throughput is the number of filtered results per unit time and total computation rate is in Giga MAC Per Second (GMACPS).

The area usage, dynamic power consumption and speed are obtained using Xilinx Timing Analyzer [74]. The throughput and computation rate are calculated according to a 1-D, 2-D and 3-D mathematical model of throughput as defined in subsections 3.2, 4.2 and 5.2 respectively. Due to the efficient implementation hierarchy of Logic fabric, signals, I/O's and hard IPs such as Block RAMs or DSP blocks, the parallel filtering architectures occupy proper logic area. Consequently, this area occupation affects the performance indices set of speed, power consumption, and throughput.

### **2.7.1 Logic Area Occupation**

The logic area usage by the multi-dimensional filtering architecture is indicated by the devices utilisation from the fabric and IPs hard cores of DSP and BRAM of the Virtex-6 FPGA board, as detailed in Table 2.1. The FPGA fabric is the underlying structure of the logic device, which consists of FFs, LUTs and Slices. The IPs hard cores are DSP48E1 and BRAM 18E1s [16, 17, 47-49].

The D-type flip-flop (FF) is the smallest storage element. The polarity of the clock (rising-edge-triggered or falling-edge-triggered) can be configured, as well as the polarity of the clock enable and set/reset signals (active-high or active-low). The total number of D-type FFs in the Virtex-6 XC6VLX240T FPGA is 301,440.

The look-up tables (LUTs) in Virtex-6 FPGAs can be configured as either one 6-input LUT (64-bit ROMs) with one output, or as two 5-input LUTs (32-bit ROMs) with separate outputs but common addresses or logic inputs. Each LUT output can optionally be registered in a flip-flop. The n-LUT can implement any possible n-input combinational logic function. Each Virtex-6 FPGA slice contains four LUTs and eight flip-flops, only some slices can use their LUTs as distributed RAM or SRLs. Each slice

has one set of clock, clock enable, and set/reset signals that are common to both logic cells. The total number of slices in the Virtex-6 XC6VLX240T FPGA is 37,680.

Virtex-6 LX240T FPGA has 416 dual-port block RAMs, each storing 36 Kbits. Each block RAM has two completely independent ports that share nothing but the stored data. Each memory access, read and write, is controlled by the clock. All inputs, data, address, clock enables, and write enables are registered. The input address is always clocked, retaining data until the next operation. An optional output data pipeline-register allows higher clock rates at the cost of an extra cycle of latency. During a write operation, the data output can reflect either the previously stored data, the newly written data, or remain unchanged.

Flexible block RAM can be configured as two 18Kb blocks or a single 36Kb block, true dual-port, simple dual-port, and FIFO. This offers independent read and write port width configuration. Xilinx claim 600MHz operation using optional pipeline capability [76]. Built-in cascade logic makes it possible to create a 64kx1 memory from two 32kx1 block RAM configurations. The aspect ratio may be configured from 32Kx1 to 1024x36 as a full dual port, or as 512x72 simple dual-port.

**Table 2.1: Characteristics of the Virtex-6 XC6VLX240T development FPGA board**

	Part Number	XC6VLX240T
	EasyPath™ FPGA Cost Reduction Solutions	XCE6VLX240T
Logic Resources	Slices	37,680
	Logic Cells	241,152
	CLB Flip-Flops	301,440
Memory Resources	Maximum Distributed RAM (Kb)	3,650
	Block RAM/FIFO w/ECC (36 Kb each)	416
	Total Block RAM (Kb)	14,976
Clock Resources	Mixed-Mode Clock Managers (MMCM)	12
I/O Resources <sup>(4,5)</sup>	Maximum Single-Ended I/O	720
	Maximum Differential I/O Pairs	360
Embedded Hard IP Resources <sup>(6)</sup>	DSP48E1 Slices	768
	PCI Express® Interface Blocks	2
	10/100/1000 Ethernet MAC Blocks	4
	GTX Low-Power Transceivers	24
	GTH High-Speed Transceivers	—
Speed Grades	Commercial	-L1, -1, -2, -3
	Extended	—
	Industrial	-L1, -1, -2
Configuration	Configuration Memory (Mb)	73.9

Virtex-6 LX240T FPGA has 768 dedicated DSP 48E1 slices operating at 600 MHz clock speed, while retaining system design flexibility. Each DSP48E1 slice fundamentally consists of a dedicated  $25 \times 18$  bit two's complement multiplier and a 48-bit accumulator (cascadable to 96 bits). Each DSP48E1 slice draws a low power consumption of only 1.09mW/100MHz at a 38% toggle rate.

### **2.7.2 Dynamic Power Consumption**

The dynamic power  $P_{dyn}$  is the dissipated power when the output changes state to charge and discharge the programmable interconnection provided for the FPGA implementation [16, 77, 78]. This occurs due to, firstly, the switching frequency of the logic area array and IOB resources. Particularly, since not every resource is toggling at the same frequency, or all the time. Secondly, short-circuit power resulting from the current flow through the transistor channels in a logic gate when all are turned on at the same time. Thus, the dynamic power is a function of the average number of outputs that are changing in each clock cycle  $\tau$ , the average node capacitance  $C$ , supply voltage  $V_{DD}$  and clock frequency  $f$ , this can be stated as;

$$P_{dyn} = \tau CV_{DD}^2 f \quad (2.1)$$

### **2.7.3 Clock Speed**

FPGAs are designed primarily as a synchronous device, thus a clock signal is required. Clock rate defines the operating speed of the algorithm implementation and is a figure measured in (MHz) that can formally quoted by FPGA vendors to give some notion of performance. The clock rates of the Xilinx Virtex 6 FPGA family is 600MHz [76]. Moreover, the inherent FPGA parallelism allows the parallel multi-dimensional data filtering architectures to maximize the amount of computation that can be performed in a single clock, which improves the performance. Thus, speed is one of the performance indices that can be improved by parallelizing the hardware.

### **2.7.4 Total Throughput**

Generally, the maximum throughput [20, 79, 80] is the maximum number of filtered results per unit time. This can be measured in Block Per Second (BPS) for the parallel 1-D signal filtering architectures; Frame Per Second (FPS) for the parallel gray image filtering architectures; Volume Per Second (VPS) for the parallel 3-D image volumetric data filtering architectures. The colour 2-D and 3-D images are measured in CFPS and CVPS respectively.

Depending on the FIR kernel size, the FPGA-based implementation is realized either by MAC convolution or FFT convolution, which affects the throughput mathematical model. The total throughputs for both are significantly improved by the use of parallelism.

### ***2.7.5 Computation Rate***

In the parallel 1-D, 2-D, 3-D and 4-D spatial convolution architectures, the total number of Multiply/Accumulate operations achieved Per Second (MACPS) can be considered as another performance index to indicate the computation rate. The total computation rate, measured in Giga MACPS (GMACPS), is directly proportional to the maximum operating clock frequency  $f$  and MAC operations accomplished by the available levels of parallelism for the 1-D, 2-D and 3-D/4-D spatial convolution architectures, as mathematically modelled in subsections 3.2.5, 4.2.6 and 5.2.6 respectively.

## **2.8 Conclusion**

In this chapter, the suitability of adapting the inherent parallelism within an FPGA to realize the hardware architectures to implement parallel multi-dimensions convolution algorithms was discussed. Three forms of parallelisms were observed in these filtering algorithms: temporal, spatial and logical parallelisms. This maps efficiently on FPGA-based architectures, by minimum flexible wordlength sized to the required arithmetic's resolution, which can be changed at different parts of inherent parallelism hardware.

The overall FPGA implementation process was depicted as a development flow of seven key steps. Parallel field programming using Xilinx System Generator was discussed as a dataflow graphical programming tool that facilitates FPGA hardware design by providing access to underlying FPGA resources. The performance indices of the parallel multi-dimensional filtering architectures were introduced as a complete package of area, speed, power, throughput and computation rate values when using XSG to target a Virtex-6 ML605 FPGA board.

# Chapter 3. Parallel 1-D Filtering Algorithm and its FPGA Implementations

## 3.1 Introduction

Most of real time 1-D filtering applications have a noisy signal to be filtered with high computing rates [25, 33, 81-86]. On the other hand, the current DSP computing technology has insufficient memory to hold the entire signal to be processed simultaneously [16, 17, 46-49, 53, 82, 87]. Thus, to achieve efficient performance results, the long real time 1-D signal is segmented so that DSP computing hardware can simultaneously and independently processes each segment in parallel. Consequently, in this chapter, a parallel solution is proposed for the algorithm level and an intrinsically parallel hardware platform is selected for its implementation. Therefore, a parallel 1-D linear filtering algorithm is proposed and implemented on a structurally parallel digital processing technology of the FPGA.

The essential characteristics of the filtered signal are stability and linearity in phase, hence the processing engine in the filtering algorithm will be a digital FIR filter. Applying a parallel 1-D FIR filtering algorithm using FPGA requires the development of a generalized mathematical model for the input signal segmentation, parallel digital convolution equation and the output filtered signal reconstruction.

Parallel linear FIR filtering algorithms can be digitally implemented either directly in the temporal domain by a MAC FIR engine, as a standard convolution, or indirectly in the frequency domain by an FFT convolution. Which convolution type is more efficient? The answer depends on the length of the FIR filter coefficients (kernels) [82]. Since the filtering time for standard convolution is directly proportional to the number of coefficients for a particular input signal, shorter FIR filter coefficients can be efficiently implemented with standard convolution. Longer filter coefficients can be efficiently implemented with FFT convolution [82, 88].

The remainder of this chapter is organised as follows: in section 3.2, the main research concepts are defined and mathematically explained. In section 3.3, the generalized parallel 1-D linear filtering algorithm is presented with its mathematical model using linear overlap-add block filtering method. In section 3.4, the three parallel 1-D temporal convolver architectures are developed, their performance indices are evaluated as a complete set of area, speed, dynamic power consumption, throughput and the

computation rate. In section 3.5, two FPGA implementations of 1-D FFT convolution are developed. Then, their performance indices are evaluated. In section 3.6, a practical example, the realization of 1-D cross-correlation for parallel 1-D match filtering algorithm. Finally, the conclusion of this chapter is presented in section 3.7.

## **3.2 Research Concepts Definitions**

This section demonstrates the main research concepts on which this chapter is based.

### ***3.2.1 Theory of Linear Block Filtering Method***

Linear block filtering method is a linear filtering technique for a real-time long sequence signal processed on a segment-by-segment basis. In real-time applications, the raw input signal is usually very long compared to the FIR filter kernel, e.g. digital speech filtering [26, 27, 89], digital music filtering [90], medical acoustic sounds[81] ...etc. Thus, to be digitally processed, this very long input signal is segmented to a fixed-length block to occupy part of the already limited memory size of any digital hardware systems, such as DSPs or FPGAs.

There are two well-known block filtering techniques for the linear convolution block filtering method, namely the overlap-add method or the overlap-save method. Which technique to be used depends on the FIR coefficients ( $M$ ) index, running from  $0$  to  $M-1$  or negative indexes are used respectively.

The overlap-add technique is based on four fundamental steps. First, decompose the input signal length into contiguous blocks of equal length segments. Second, filter each of the blocks individually using the FIR kernel of  $M$  coefficients, after  $(M-1)$  zero padding to the right of each block to allow for the expansion during the convolution. Third, overlap the  $M-1$  expansion results in the output filtered blocks to each other. Fourth, reconstruct the final output signal by adding these output blocks.

The key to this technique is how the lengths of these signals are affected by the convolution. When a  $P$  sample signal is convolved with an  $M$  sample filter kernel, the output signal will be expanded by  $M-1$  points to the right to be  $P+M-1$  samples long, if the filter kernel runs from index  $0$  to  $M$ . However, if negative indexes are used in the filter kernel, the expansion will be to the right and left. The overlap-add techniques applied in this chapter because of the FIR kernel index is running from  $0$  to  $M-1$ .

### 3.2.2 Linear-Phase 1-D FIR Digital Filter

The filtering function for a linear 1-D filter is an accumulated sum of point by point multiplications of a 1-D FIR coefficients,  $h(m)$  of size  $M$ , and the sample values of the input signal,  $x(n)$  of size  $N_I$ , within the window. The FIR filter interacts with signal through a process called linear convolution [46], which is represented by:

$$y(n) = x(n) * h(m) \quad (3.1)$$

where,  $*$  is the linear 1-D convolution. This convolution process is formally defined by:

$$y(n) = \sum_{m=0}^{M-1} x(m)h(n - m) \quad (3.2)$$

This linear filtering process is equivalent to performing a 1-D convolution with the coefficients flipped left-for-right. Thus, a windowed-base FIR filter can be designed to be a linear phase by making its impulse response coefficients have left-right symmetry to achieve this flip. Thus, a linear phase 1-D MAC FIR filter can achieve 1-D convolution.

Parallel 1-D convolution audio filtering algorithms use a lowpass linear phase FIR filter, which maps well onto either a 1-D multiply-accumulate (MAC) engine or 1-D FFT convolution unit. In general,  $M$  MAC operations are required to compute an output sample for an  $M$ -tap filter. The  $M$ -tap FIR filter may be implemented on more than one unique MAC unit taking into consideration the tradeoffs among filter performance indices, such as area, speed, dynamic power and throughput. Hence, three configurations are developed for a single MAC, dual MAC and quad MAC FIR units, which are shown in Figure 3.4, Figure 3.5 and Figure 3.6 respectively.

### 3.2.3 1-D FFT Convolver Engine

1-D Convolution in the time domain, as stated in (3.2), corresponds to multiplication of the two spectra of the 1-D FIR filter and the raw input signal. Multiplication costs less logic area in digital implementation than convolution. This Fourier property enables 1-D FFT convolution to be utilized in linear signal filtering, as shown in Figure 3.1.

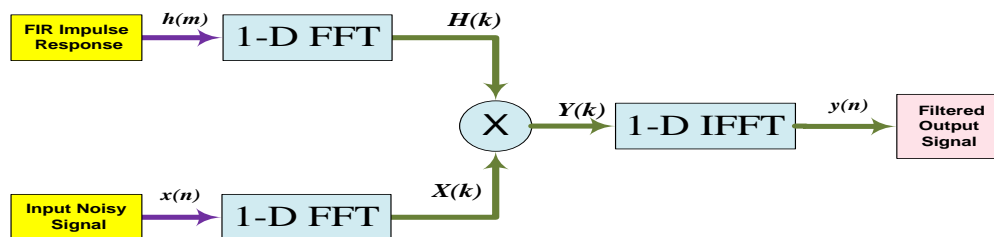


Figure 3.1: 1-D FFT convolver unit

The FFT converts the input segment and the FIR kernel into the real and imaginary parts of the frequency response. The frequency spectrum of the output segment  $Y(k)$  is then found by multiplying the FIR kernel frequency response  $H(k)$  by the spectrum of the input segment  $X(k)$ . Since, these spectra consist of real and imaginary parts, then a complex multiplication is carried out in rectangular form as follows;

$$Y_r(k) = X_r(k) \times H_r(k) - X_i(k) \times H_i(k) \quad (3.3)$$

$$Y_i(k) = X_i(k) \times H_r(k) + X_r(k) \times H_i(k) \quad (3.4)$$

The complex multiplication of the above two equations can be built up from four real multiplications and two real additions [16, 91].

The number of complex multiplications and additions required for a radix-2 1-D FFT algorithm of length  $N_1$  are  $(\frac{N_1}{2}) \log_2 N_1$  and  $N_1 \log_2 N_1$ , respectively. Therefore, the FFT reduces the computational complexity from  $O(N_1M)$  to  $O(N_1 \log_2 N_1)$ . This considerable gain in computation, particularly for a thousand long FIR kernel, justifies the hardware implementation of the FFT over the direct MAC FIR filter.

### 3.2.4 Total Throughput

Generally, the maximum throughput [20, 79, 80] is the maximum clock frequency divided by one sample block, measured in BPS, at a filtering rate of one sample per clock cycle. The throughput of the parallel 1-D MAC FIR filtering architecture is limited by the 1-D MAC FIR operation for a particular signal, which can be mitigated by parallelism. Thus, the total throughput is directly proportional to the levels of parallelism and limited by the 1-D MAC FIR engine throughput  $\mu_{FIR}$ . The effective levels of parallelism are the number of parallel 1-D direct convolver filters ( $\alpha_1$ ) and the number of multi-MAC engines ( $\alpha_2$ ) per 1-D convolver unit. Then, the total throughput  $\mu$  can be formulated as:

$$\mu = \frac{\mu_{FIR}}{N_1} \times \alpha_1 \quad (3.5)$$

where,  $N_1$  is the size of the input signal's block.

The 1-D MAC FIR unit throughput  $\mu_{FIR}$  is directly proportional to the clock speed  $f$  and limited by the 1-D FIR coefficients ( $M$ ) which can be partitioned to be processed by multi-MAC engines. Then  $\mu_{FIR}$  can be stated as:



$$\mu_{FIR} = \left(\frac{f}{M}\right) \times \alpha_2 \quad (3.6)$$

Thus, the total throughput for the parallel 1-D MAC FIR filtering architecture can be defined as:

$$\mu = \frac{f \alpha_1 \alpha_2}{N_1 M} \quad (3.7)$$

where,  $\frac{N_1 M}{\alpha_1 \alpha_2}$  is one sample block at a filtering rate of one sample per clock cycle.  $\alpha_1$  is the number of input signal partitioned sub-blocks, here,  $\alpha_1=4$ , then, one sample block length,  $\frac{N_1}{\alpha_1} = \frac{110032}{4} = 27508$  samples per one input block for a real-time (22050 KHz /1Ch/16 bit) speech signal.  $\alpha_2$  is the degree of parallelism inside the 1-D temporal convolver unit, using single, dual or quad MAC engines, then,  $\alpha_2 = 1, 2$  or  $4$  respectively.

For a parallel 1-D FFT filtering architecture, a one sample block time is  $\frac{N_1}{\alpha_1} \log_2 \frac{N_1}{\alpha_1}$ . Because a single FFT convolution unit in the parallel FFT algorithm performs the convolution (3.2) using (3.8) in  $O\left(\frac{N_1}{\alpha_1} \log_2 \frac{N_1}{\alpha_1}\right)$  steps [44], where,  $N_1$  is the length of the real-time (22050 KHz /1Ch/16 bit) speech signal. While, temporal convolution requires  $O\left(\frac{N_1 M}{\alpha_1 \alpha_2}\right)$  operations, this implies a gain, since  $\left(\frac{M}{\alpha_2}\right) > \log_2 \left(\frac{N_1}{\alpha_1}\right)$  at  $M > 64$  to apply the FFT convolution:

$$x * h(m) \equiv F^{-1}(F(x) \times F(h)) \quad (3.8)$$

Where,  $F^{-1}$ ,  $F$  and  $\times$  are the Fourier transform, the inverse Fourier transform, and pointwise complex multiplication, respectively.

Thus, the overall throughput  $\mu$  of the parallel FFT block filtering architecture of  $\alpha_1$  parallel processing stages and maximum clock frequency  $f$  (MHz) can be formulated as:

$$\mu = \frac{f}{\frac{N_1}{\alpha_1} \log_2 \frac{N_1}{\alpha_1}} \quad (3.9)$$

### 3.2.5 Total Computation Rate

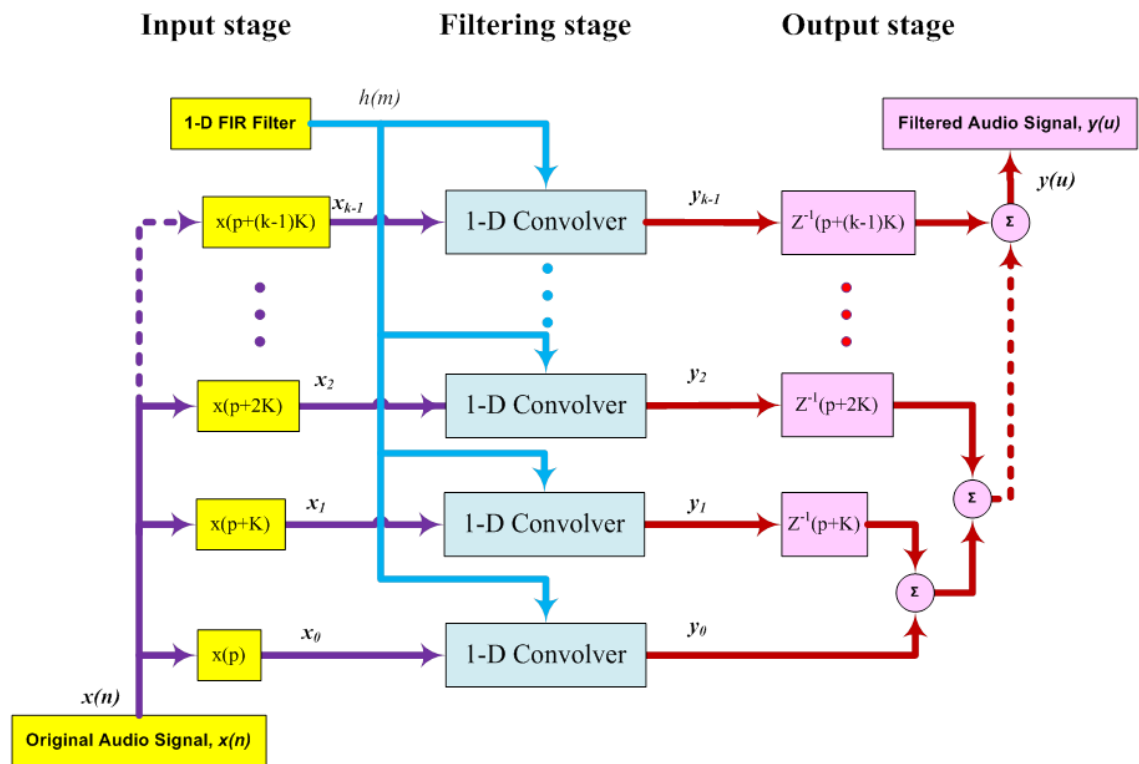
The total number of MACPS can be considered as another performance index to indicate the computation rate for each one of the three 1-D direct convolution architectures. Where, the total computation rate  $\gamma$ , measured in Giga MACPS (GMACPS), is directly proportional to the maximum operating clock frequency  $f$ , the

number ( $\alpha_1$ ) of parallel 1-D direct convolver stages and the number ( $\alpha_2$ ) of parallel MAC engines per 1-D direct convolver, thus

$$\gamma = f \times \alpha_1 \times \alpha_2 \quad (3.10)$$

### 3.3 The Generalized Parallel 1-D Linear Block Filtering Algorithm

The generalized parallel 1-D filtering algorithm, as shown in Figure 3.2, is presented for the overlap-add block filtering method. This linear block filtering algorithm consists of three stages: serial to parallel input stage, parallel filtering processing stage and parallel to serial output stage.



Segmentation length:  $0 \leq p \leq K-1$ , where,  $K = \frac{\text{input length } (N)}{\text{number of segments } (k)}$ , Filtered output length:  $u = P + M - 1$

**Figure 3.2: The Generalized Parallel 1-D Linear block Filtering Algorithm**

The three stages are implemented on the Virtex-6 ML605 development board using XSG. The mathematical model of these three stages are developed and expressed in the following sections.

#### 3.3.1 Input Segmentation Stage

The input signal segmentation of Figure 3.2 equally divides a long input sequence  $x(n)$  of size  $P$  into non-overlapping subsections ( $x_0(p), x_1(p), x_2(p), \dots, x_{k-1}(p)$ ) each of size

$\left(\frac{P}{k}\right)$ . Thus, the generalized mathematical formula that model the input segmentation stage can be expressed as:

$$x(n) = x_0(p) + x_1\left(p + \frac{P}{k}\right) + \dots + x_{k-1}\left(p + (k-1)\frac{P}{k}\right), \quad 0 \leq p \leq \frac{P}{k} - 1 \quad (3.11)$$

This can be expressed in a closed form as:

$$x(n) = \sum_{r=0}^{k-1} x_r\left(p + (r)\frac{P}{k}\right), \quad 0 \leq r \leq k-1, \quad \text{and } 0 \leq p \leq \frac{P}{k} - 1 \quad (3.12)$$

where  $n$  is the input size of  $0 \leq n \leq P-1$ . Each term in the right hand side of (3.11) and (3.12) is a separate segment of the input signal,  $x(n)$ , to be independently filtered by a separate 1-D convolution unit (convolver) in the Virtex-6 board.

### 3.3.2 Parallel Filtering Stage

In Figure 3.2, the parallel filtering stage is implemented as an array of parallel 1-D convolution units (convolver). The filtering sub-signal  $y_r$  can be represented in the time domain as the linear convolution of the sub-segment  $x_r$  sequences over the FIR impulse response  $h(m)$  sequence, as:

$$y_r = \sum_{n=0}^{p-1} x_r(n + rp) h(m - n) \quad (3.13)$$

The resultant filtering is the sum of all the parallel sub-filters, and can be formally, expressed as:

$$y(u) = \sum_{r=0}^{k-1} \left( \sum_{n=0}^{p-1} x_r(n + rp) h(m - n) \right) \quad (3.14)$$

where,  $0 \leq u \leq P + M - 1$  is the final output length,  $k$  is the number of segments and  $p$  is the segment length. The filtering stage is computationally intensive. Thus, the number of segments ( $k$ ) and the impulse response size  $h(m)$  have to be optimized so that the number of the 1-D convolver units is minimum to achieve the required level of performance [72].

### 3.3.3 Output Reconstruction Stage

In Figure 3.2, the pipelined output  $y(u)$  is reconstructed from the parallel filtered sub outputs ( $y_0(n), y_1(n), \dots, y_{k-1}(n)$ ) of the parallel 1-D convolution units, by an adder tree. Thus, the 1-D output formula is:

$$y(u) = y_0(n) + y_1\left(n + \frac{N}{k}\right) + \dots + y_{k-1}\left(n + (k-1)\frac{N}{k}\right), \quad 0 \leq n \leq \frac{N}{k} - 1 \quad (3.15)$$

This can be expressed in a closed form as:

$$y(u) = \sum_r^{k-1} y_r\left(n + (r)\frac{N}{k}\right), \quad 0 \leq r \leq k-1 \text{ and } 0 \leq u \leq N + M - 1 \quad (3.16)$$

Each term in the right hand side of (3.15) and (3.16) is an output of a separate 1-D convolver. These sub-outputs are overlapped by the size of the impulse response filter length ( $M-1$ ), which is added to the next block to reconstruct the final filtered output.

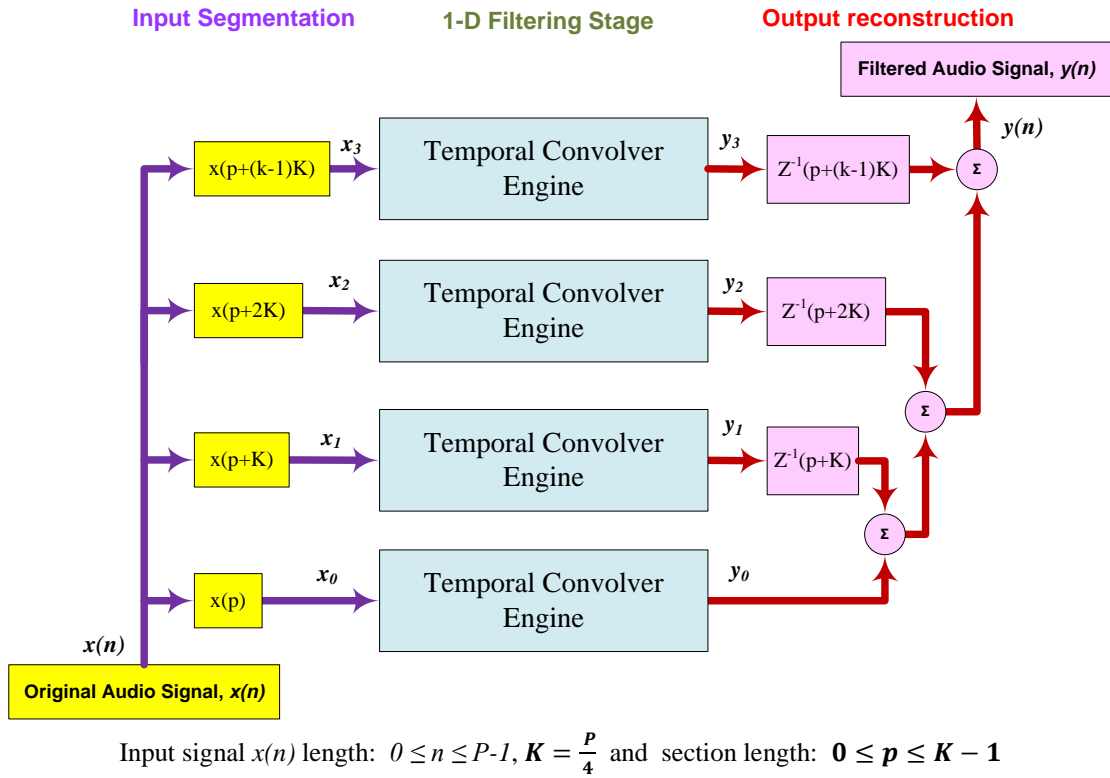
The generalized parallel 1-D linear block filtering algorithm can be realized in hardware as either a temporal convolution in the time domain by MAC FIR filters or, indirectly in a spectral domain by an existing fast transform, that depends on the filter kernel length. The parallel algorithm can directly be implemented faster by the MAC FIR convolution for the FIR kernel length shorter than (64)[82], and the filtering time is proportional to the kernel length. However, longer filter kernels can efficiently be implemented faster with the spectral convolution. With an FFT convolution, the filter kernel can be made as long as necessary for the application and the hardware platforms can handle. The temporal filtering is presented in subsection 3.4, and the spectral filtering is presented in subsection 3.5.

### 3.4 Parallel 1-D Temporal Convolver Architectures

The generalized parallel 1-D linear filtering algorithm can be realized by temporal convolution using MAC FIR filter units as shown in the general MAC FIR architecture of Figure 3.3. Consequently, the real-time 1-D filtering applications in which the FIR kernel length shorter than 64 coefficients can be achieved without the need to pad zeros to the input segments, since the 1-D FIR kernels are stored in the multi-MAC engines.

In the input stage, the 1-D signal data samples, length  $P$ , of a real-time (22050 KHz /1Ch/16 bit) speech signal are equally segmented into four blocks  $x_0, x_1, x_2$  and  $x_3$ . Each block is simultaneously and independently streamed into the 1-D multi-MAC convolver engine, as shown in Figure 3.3.

The 1-D filtering stage of this generic architecture can be realized by more than one unique implementation depending on the parallelism style inside the MAC FIR engine, in addition to the main parallelism of four parallel data paths. This inside parallelism style is either a single, dual and quad MAC FIR units in the processing stage, which is shown in Figure 3.4, Figure 3.5 and Figure 3.6 respectively.



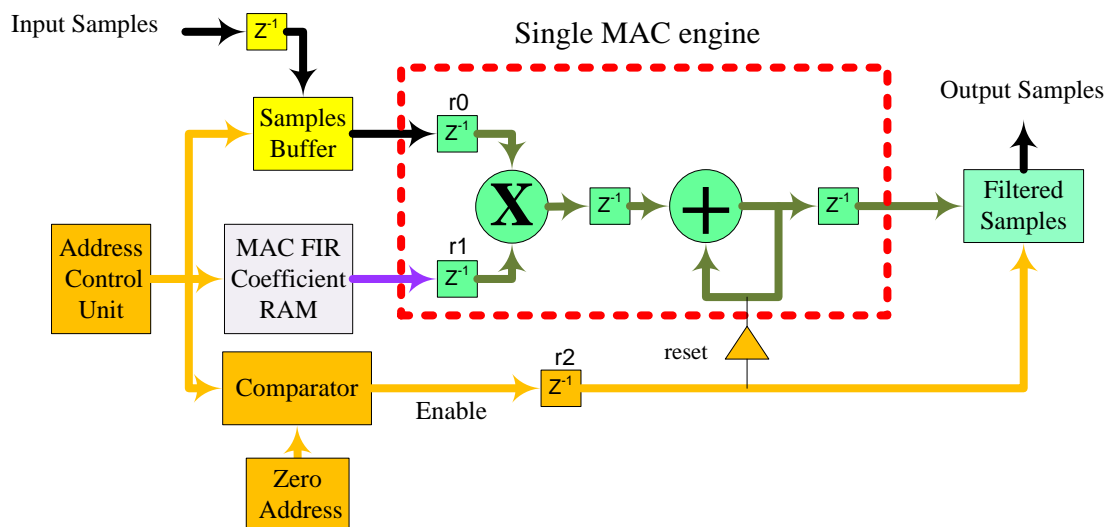
**Figure 3.3: Parallel 1-D multi-MAC filtering architecture**

The output stage is a parallel to serial pipelined process of reconstruction, recording and displaying the filtered real-time speech output. The speech reconstruction is achieved using three pipelined adders tree. The pipelining of the adder tree is sized according to the overlap-add technique, as shown in Figure 3.3. The final reconstructed output is connected to a gateway-out block to convert the fixed point filtered samples to floating point numerical representation used by Simulink blocks for recording as a WAV file and displaying the filtered real-time speech signals. Consequently, the parallel 1-D filtering algorithm can be implemented by three temporal architectures according to the hardware structure of their 1-D convolver engine. The input stage and the output stage are the same for the three architectures, thus its convolver engine will characterize each architecture.

### 3.4.1 Parallel Single MAC Convolver Architecture

Figure 3.4 shows unit1, a single MAC convolver of architecture1. This parallel architecture is an implementation for the parallel linear FIR filtering algorithm of Figure 3.3. Unit 1 consists of input buffering components, the single MAC FIR engine, control circuitry, and filtered output buffer.

Input buffering components consist of an addressable shift register (ASR) and a single port RAM. The ASR and RAM act as the input sample buffer and filter coefficients storage respectively. The ASR address port runs  $M$  times faster than the data port. The RAM is configured to use block memory. The multiply-accumulate engine is implemented using a dedicated DSP 48E1s multiplier block and accumulator block. A single counter implements the control circuitry as a free running counter from  $0$  to  $M-1$ , and then repeats. A capture register act as the filtered output buffer, required for streaming operation, since the MAC engine reloads its accumulator with an incoming sample after computing the last partial product for an output sample.



**Figure 3.4: Unit 1; 1-D Single MAC convolver implementation**

A counter generates the RAM and ASR addresses. For every new input sample, the accumulator block is reset to its current input, and the capture register latches the MAC engine output. This occurs when a comparator generates the reset and enables pulses for the accumulator and capture register. The pulse is asserted when the address is zero and is delayed to account for pipeline stages. Pipeline registers  $r_0$ ,  $r_1$  and  $r_2$  are included to increase performance. Upon reset, the accumulator re-initializes to its current input value rather than zero, which allows the MAC engine to stream data without stalling for one sample period.

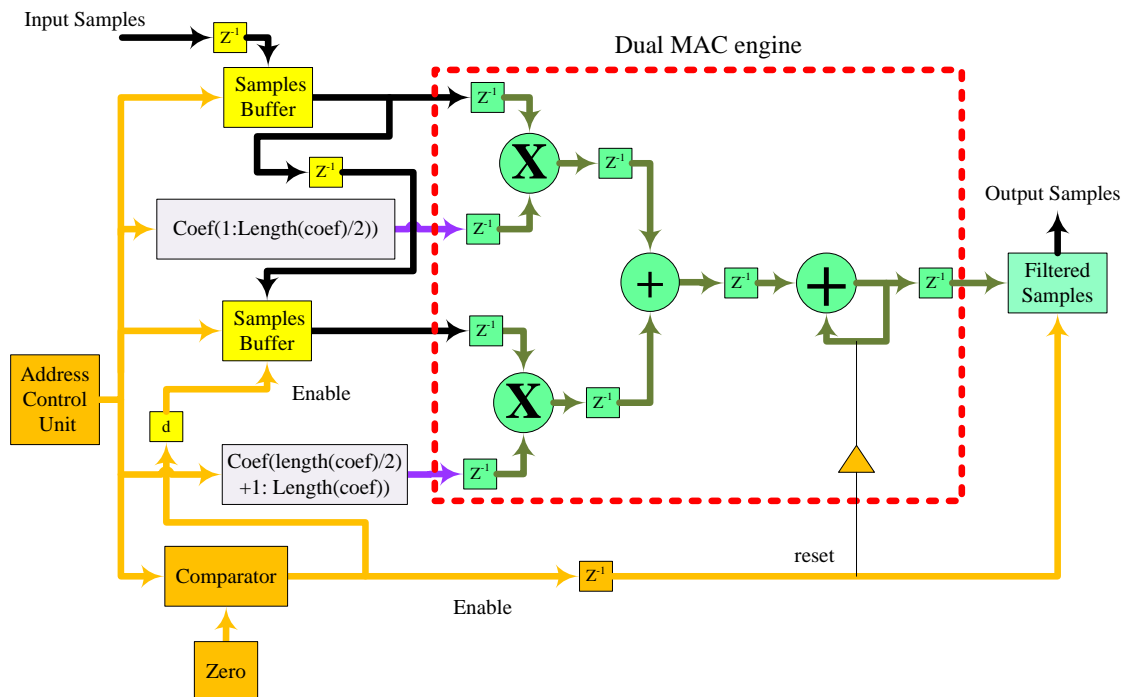
As shown in Table 3.1, the single MAC 1-D convolver occupies less logic area than either the dual MAC 1-D or the quad MAC 1-D convolver. Unit 1 of the single MAC 1-D convolver occupies less logic area than unit 2 by (27%) FFs, (31%) LUTs, (20%) slices, (100%) dedicated DSP 48E1s multiplier and RAMB 18E1s block memory. In addition, compared to unit 3, unit 1 occupies less logic area by (52%) FFs, (59%) LUTs, (48%) slices, (200%) dedicated DSP 48E1s multiplier/ RAMB 18E1s block memory.

**Table 3.1: Logic Devices utilization by the three 1-D temporal convolver units**

	Single MAC 1-D convolver unit	Dual MAC 1-D convolver unit	Quad MAC 1-D convolver unit
	Unit 1	Unit 2	Unit 3
FFs	146	201	302
LUTs	90	130	221
Slices	158	198	302
DSP 48E1s	1	2	4
RAMB 18E1s	1	2	4

### 3.4.2 Parallel Dual MAC Convolver Architecture

Figure 3.5 shows unit 2, a dual MAC convolver engine of architecture 2. This parallel architecture is another implementation for the parallel linear FIR filtering algorithm of Figure 3.3 .



**Figure 3.5: Unit 2; Implementation of 1-D Dual MAC convolver**

Unit 2 of the dual MAC 1-D convolver consists of cascaded buffering components, parallel coefficients block RAMs pair, control circuitry and a dual MAC FIR engine.

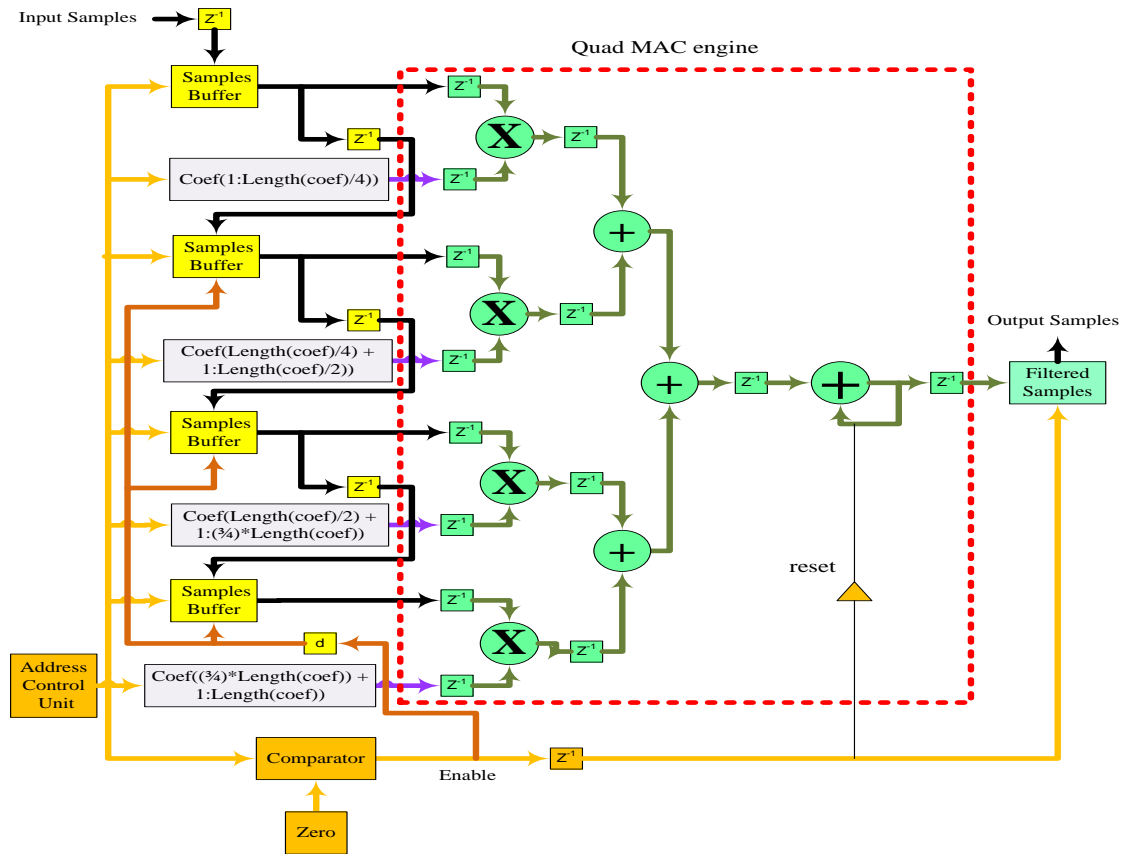
Two ASRs are cascaded to form the sample input buffer. ASR2 is enabled when the last element of ASR1 is addressed to ensure that the sample data is correctly propagated between ASRs. The filter coefficients are equally stored in two block RAMs. RAM1 stores the first half of the coefficients and RAM2 stores the remaining coefficients, as shown in Figure 3.5. Both RAMs and ASR pairs share the same address sequence generated by the counter. The sequence counts from 0 to  $(\frac{M_1}{2} - 1)$ , and then repeats. A delayed reset/enable pulse generated by the comparator drives the enable port of ASR2. Block *d* delays the signal, and is parameterized with the appropriate latency of  $(\frac{length(coef)}{2} - 2)$  using the coefficient array length. The filter reduces the number of MAC operations required to compute the sum of products by distributing the workload between 2 parallel multipliers. The products are added and the sum is accumulated.

### ***3.4.3 Parallel Quad MAC Convolver Architecture***

Figure 3.6 shows unit 3, a quad Mac convolver of architecture 3. This parallel architecture is a third implementation for the parallel linear FIR filtering algorithm of Figure 3.3. Unit 3 consists of four cascaded buffering components, four parallel coefficients block RAMs, control circuitry and a parallel MAC FIR engine.

Four ASRs are cascaded to form the sample input buffers. ASR2, ASR3 and ASR4 are enabled when the last element of ASR1 is addressed to ensure that the sample data is correctly propagated between ASRs. The filter coefficients are equally stored in four block RAMs, each of the RAMs storing a quarter of the coefficients, as shown in Figure 3.4. Both RAMs and ASRs blocks share the same address sequence generated by the counter. The sequence counts from 0 to  $(\frac{M_1}{4} - 1)$ , and then repeats. A delayed enable pulse generated by the comparator drives the enable port of ASR2, ASR3 and ASR4. Block *d* delays the signal, and is parameterized with the appropriate latency of  $(\frac{length(coef)}{4} - 2)$  using the coefficient array length. The filter reduces the number of MAC operations required to compute the sum of products by distributing the workload between four parallel multipliers. The products are added in an adder tree, and the sum is accumulated. Although the positioning of the adder tree and accumulator are interchangeable, placing the adder tree before the accumulator results in a resource optimized implementation.





**Figure 3.6: Unit 3; 1-D Quad MAC convolver implementation**

### 3.4.4 Performance Indices of the Three Parallel Multi-MAC Convolver Architectures

The performance indices of the three 1-D temporal convolver architectures are considered as a complete set of area, speed, power, throughput and computation rate performance parameters using XSG to target a Virtex-6 ML605 board. The minimized utilized area of the three architectures, as shown in Table 3.2, are due to the efficient implementation hierarchy of logic fabric, signals, I/O's and hard IPs such as Block RAMs or DSP blocks. These three architectures occupy proper logic area of FFs, LUTs and slices. Where, architecture 3 and 2 are occupying, in average, more than double and less than 1.5 logic area that of architecture 1 respectively. Additionally, architecture 3 and 2 are respectively using four and two times the number of multipliers / block RAMs Hard IPs for its MAC engines than that of architecture 1. Consequently, this area occupation affects the performance indices of speed, power consumption, throughput and computation rate as shown in Table 3.3.

Several observations can be made from Table 3.3. Firstly, the operating clock frequency is particularly insensitive to the number of MAC convolver engines for the three architectures, and principally operating around the maximum frequency of 225 MHz

[31]. Secondly, the dynamic power consumption at (40 nm) junction temperature of 54°C decreases, monotonically, from 12mW, 18mW and 24mW down to 5mW, 8mW and 13mW for architectures 1, 2 and 3 respectively as the MAC FIR coefficients length descends from 64 to 2 coefficients. Thirdly, due to parallelism style, the highest throughput is achieved by architecture 3, almost double and four times that of architecture 2 and 1 respectively. As may be predicted by (3.7), where, the architecture throughput is a function of the maximum operating frequency and the one input sample block. Fourthly, the highest computation rate is accomplished by architecture 3, double and four times greater than that of architecture 2 and 1 respectively; this is predicted from (3.10), the computation rate is affected by the two levels of parallelism inside the MAC engines and the concurrent filtering stages.

**Table 3.2: Logic Devices utilization by the three parallel temporal convolver architectures**

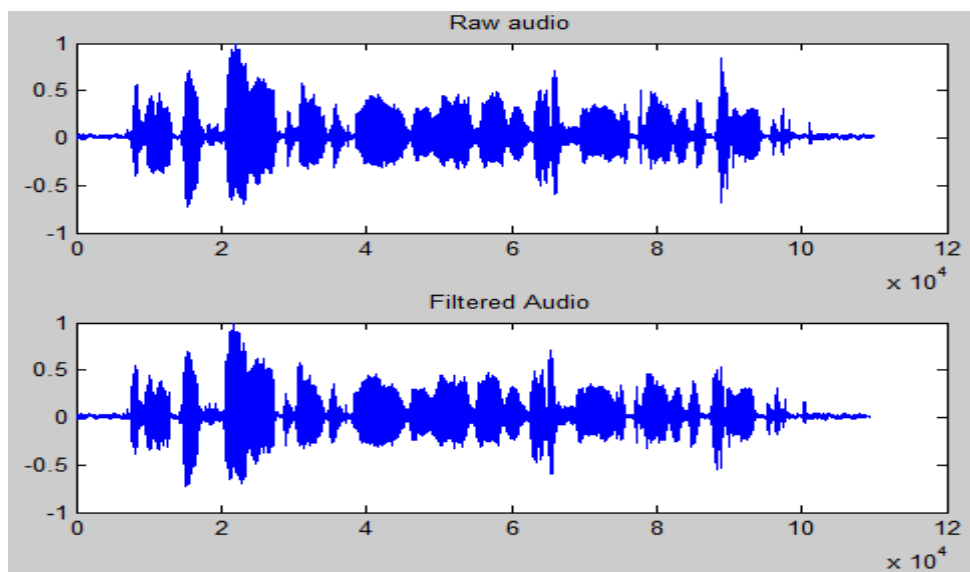
	Architecture 1 (Single MAC convolvers)							Architecture 2 (Dual MAC convolvers)					Architecture 3 (Quad MAC convolvers)			
	FIR impulse Response length															
	2	3	5	7	15	31	63	4	8	16	32	64	8	16	32	64
<b>FFs</b>	444	496	501	513	518	523	528	675	703	708	713	734	1092	1017	1112	1129
<b>LUTs</b>	283	318	334	341	362	391	414	509	524	541	549	557	911	919	929	974
<b>Slices</b>	151	158	162	175	184	186	243	262	268	272	275	291	381	390	396	409
<b>DSP 48E1s</b>	4	4	4	4	4	4	4	8	8	8	8	8	16	16	16	16
<b>RAMB 18E1s</b>	4	4	4	4	4	4	4	8	8	8	8	8	16	16	16	16

**Table 3.3: Performance indices of the parallel 1-D multi-MAC convolver filter architectures**

FIR Filter kernel ( <i>M</i> )	Maximum Clock Frequency (MHz)			Dynamic Power (mWatt)			Throughput (BPS)			computation rate (GMACPS)		
	<i>Arcit</i> 1	<i>Arcit</i> 2	<i>Arcit</i> 3	<i>Arcit</i> 1	<i>Arcit</i> 2	<i>Arcit</i> 3	<i>Arcit</i> 1	<i>Arcit</i> 2	<i>Arcit</i> 3	<i>Arcit</i> 1	<i>Arcit</i> 2	<i>Arcit</i> 3
<b>2</b>	225			5			4090			0.9		
<b>3</b>	225			5			2726			0.9		
<b>4</b>		225			8			4090				
<b>5</b>	225			5			1636			0.9		
<b>7</b>	225			6			1168					
<b>8</b>		225	223		9	13		2045	4090	0.9	1.8	3.6
<b>15</b>	224			7			618					
<b>16</b>		222	225		13	17		1022	2045	0.9	1.8	3.6
<b>31</b>	225			9			264					
<b>32</b>		225	225		15	19		511	1022	0.9	1.8	3.6
<b>63</b>	223			12			130					
<b>64</b>		224	222		18	24		256	511	0.9	1.8	3.6

For comparison purposes, Hwang and Ballagh [31] carried out a research work on the implementations of FIR filters using System Generator, the trade off between filter size and throughput was discussed by providing the performance results of three 64-tap FIR filters with a varying number of MAC engines. The performance indices of the occupied slices and speed were taken into consideration. The operating clock frequency was not particularly sensitive to the number of MAC-engines employed. A single-MAC architecture has the drawback that the throughput is inversely proportional to the number of filter taps. The throughput can be increased dramatically by exploiting parallelism that matches resource usage and availability to throughput, using System Generator.

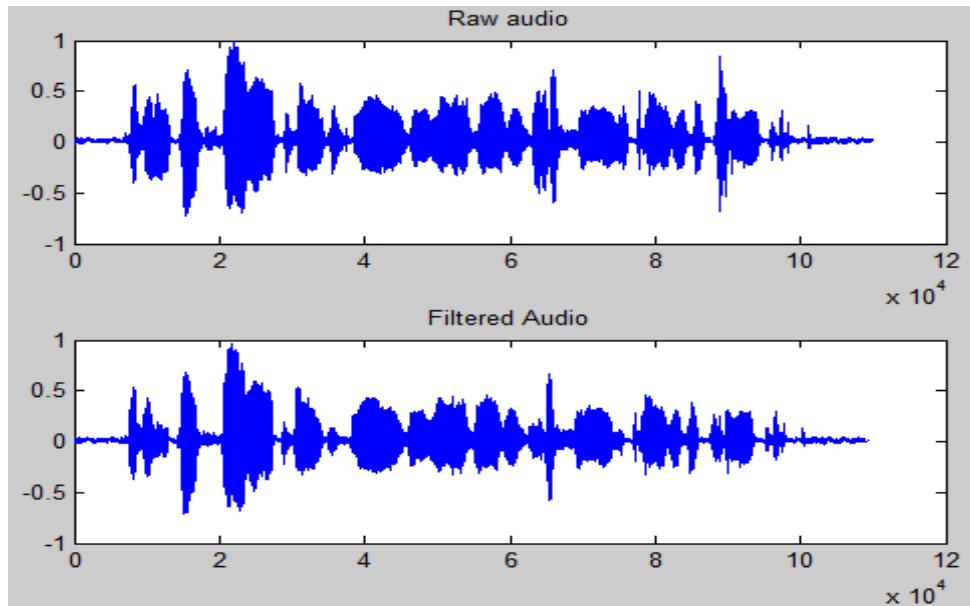
In one dimensional filtering , the linear phase FIR filter, as defined in subsection 3.2.2, can be designed as a low-pass window-base FIR, using Xilinx FDATool block, for filter design and analysis tool. The three implementations of the parallel temporal filtering algorithm are developed as “plug and filter” architectures. The 1-D FIR filter can be plugged and developed to filter an input signal. The filtering results produced using architectures 1, 2 and 3 are shown in Figure 3.7, Figure 3.8 and Figure 3.9 respectively.



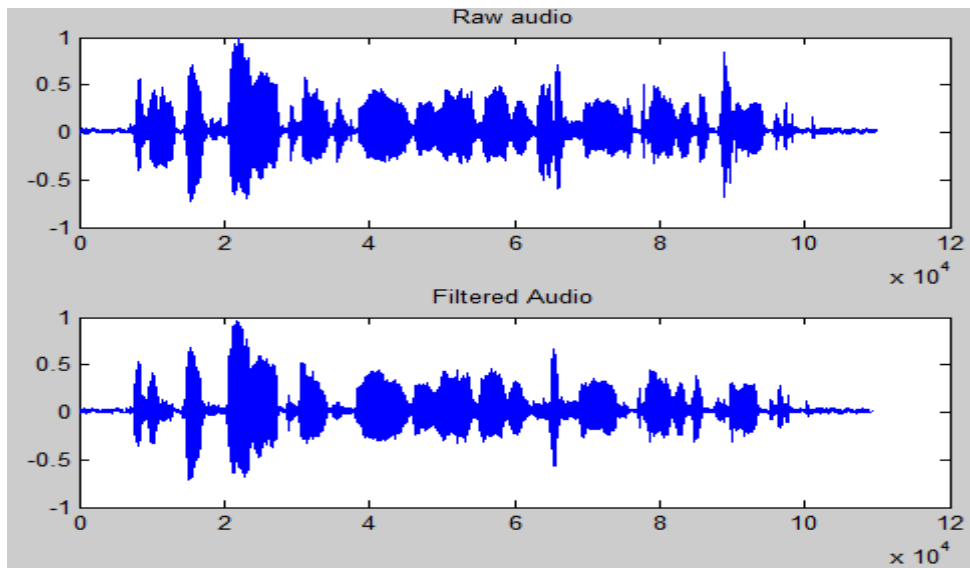
**Figure 3.7: Architecture1’s filtering results of a raw real-time speech (22050 Hz /1Ch/16 bit) signal at 1-D FIR kernels of 2 coefficients**

At the top of each figure, the speech signatures of the noisy real-time speech (22050 Hz /1Ch/16 bit) signal to be compared with the filtering results. Figure 3.7 shows Architecture1 filtering results for 1-D single MAC FIR kernels at 2 coefficients. Figure 3.8 shows Architecture 2 filtering results of 1-D dual MAC FIR kernels that can be divided by two and more than one coefficient per MAC engine at 4 coefficients. Figure

3.9 shows Architecture3 filtering results using quad MAC FIR kernels that can be divided by four and more than one coefficient per MAC engine at 8 coefficients. All the filtered speech signals are enhanced.



**Figure 3.8: Architecture2's filtering results of a raw real-time speech (22050 Hz /1Ch/16 bit) signal at 1-D FIR kernels of 4 coefficients.**



**Figure 3.9: Architecture3's filtering results of a raw real-time speech (22050 Hz /1Ch/16 bit) signal at 1-D FIR kernels of 8 coefficients.**

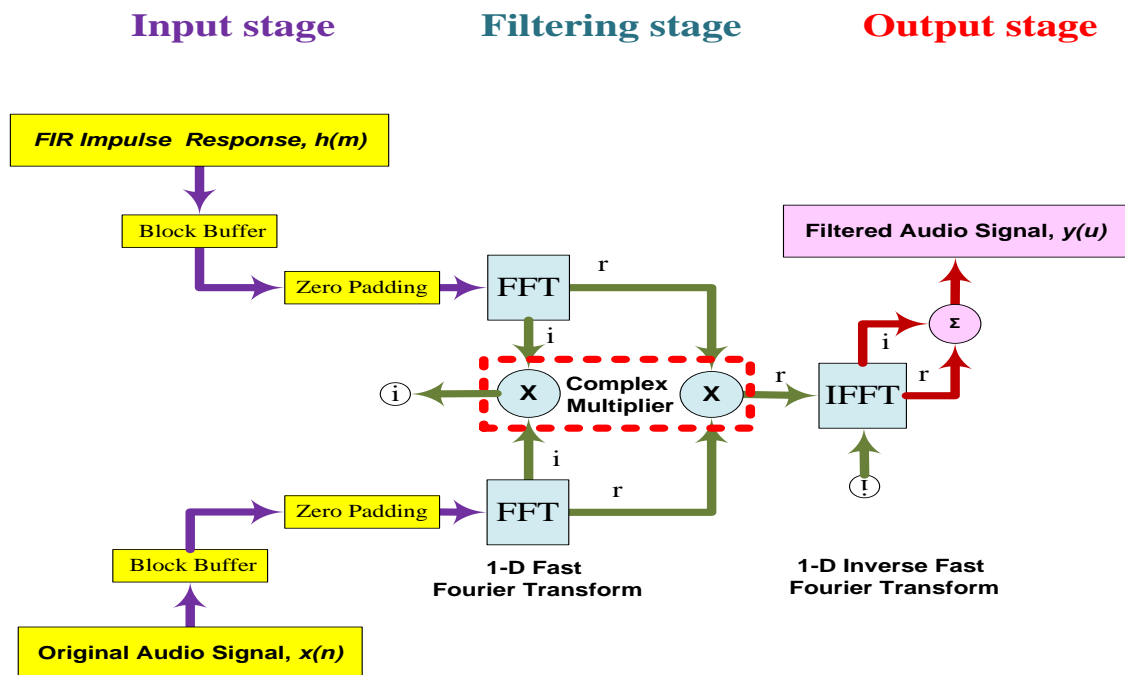
### 3.5 Parallel 1-D FFT Convolver Architectures

To cover the real-time linear 1-D filtering applications [26, 27, 81, 92] in which the FIR kernel length is longer than 64 coefficients, the parallel 1-D filtering algorithm's FPGA implementation can be efficiently captured by more than one fast parallel 1-D FFT filtering architecture, depending on the abstraction level of the implementation and the optimization approach for the performance efficient indices. Consequently, two generic

architectures are developed to implement in Virtex-6 board; the single 1-D FFT filtering unit (convolver engine) and the parallel 1-D FFT block filtering algorithm. They are described in subsections 3.5.1 and 3.5.2 respectively.

### 3.5.1 Parallel 1-D FFT Convolver Architecture

Architecture 4, as shown in Figure 3.10, realizes a single 1-D FFT convolver unit achieving high-speed filtering performance by utilizing the FFT convolution's principle; convolution in the time domain corresponds to multiplication in the frequency domain [25]. This architecture consists of input, filtering and output stages.

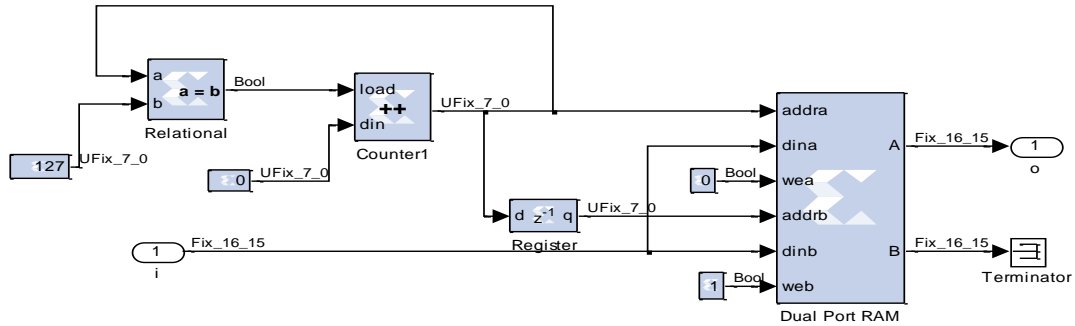


**Figure 3.10: Architecture 4; implementation of fast single 1-D FFT convolver unit**

Within the input stage the 1-D signal and the 1-D FIR filter are streamed, and zero padded to be equal in size to the N-point of the FFT. Both streams are block buffered to be processed in the filtering stage. The block buffer, as shown in Figure 3.11, can be implemented as a dual port RAM and addressing control circuit.

The dual port RAM, RAMB18E1s, has two independent sets of ports for simultaneous reading and writing. Independent address, data, and write enable ports allow shared access to a single memory space. Hence, each port set has one output port and three input ports for address, input data, and write enable. When the RAM word depth is longer than the input segment, then the RAM's trailing words are set to zero. Thus, this block buffer can accomplish the zero padding function without using extra hardware.

Since,  $N=p+M-1$ , then, by setting the RAM depth to  $N$  and the buffer depth to  $p$ , the dual port Ram also accomplishes the zero padding function.



**Figure 3.11: Implementation of the Block Buffer component**

The RAM's addressing control circuit is an up counter to the RAM depth size compared against the buffered block size. This address generation is used when writing data samples and the reading of the stored samples data.

The 1-D filtering stage is a temporal to spectral transformation structure, as explained in subsection 3.2.3. This transformation structure consists of a 1-D FFT, a complex multiplier and a 1-D IFFT blocks. The 1-D FFT can be realized by Xilinx FFT v7\_0 blocks to produce the complex  $X(k_l)$ . Similarly, 1-D FFT calculates the 1-D FIR filter's frequency spectrum, that is the real and imaginary parts of  $H(k_l)$ . Since, these spectra consist of real and imaginary parts, then, a frequency domain multiplication is carried out in rectangular form according to (3.3) and (3.4) Using Xilinx complex multiplier block, these two frequency spectra are point-by-point multiplied to produce  $Y(k_l)$ , which represents the 1-D FFT of the filtered image. To transform back into the time domain, the inverse Fourier transform of  $Y(k_l)$  is calculated by taking the 1-D inverse FFT.

In the output stage, the two streams of the filtered signal is summed to produce the 1-D filtered signal  $y(n_l)$ . The final reconstructed output is connected to a gateway-out block, which provides the conversion from the fixed point format which is used by the FPGA to floating point numerical representation used by Simulink blocks for displaying the filtered MRI.

### 3.5.2 Fast Parallel 1-D FFT Convolver Architecture

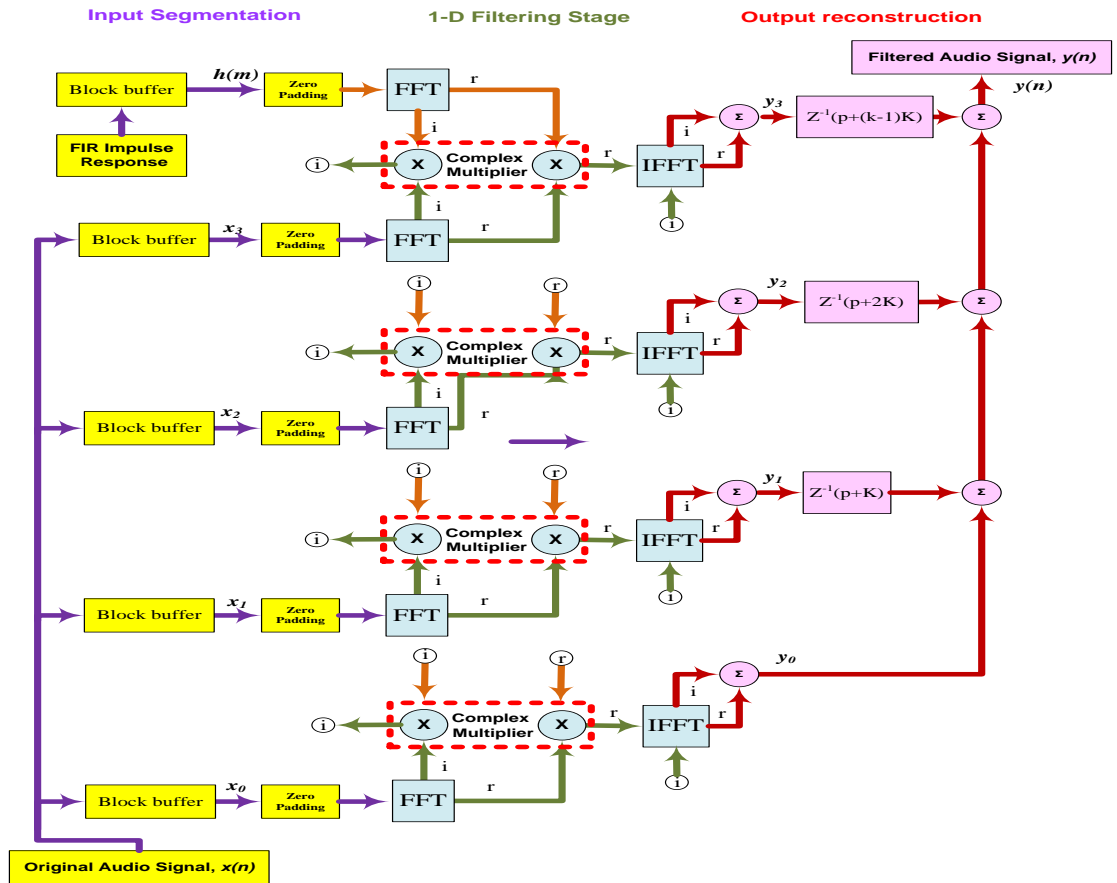
Architecture 5, as shown in Figure 3.12, implements a parallel 1-D FFT convolver structure using the overlap-add block filtering technique as defined in sub-section 3.2.1. In the input stage, the 1-D signal data samples, length  $P$ , of the real-time speech signal (22050 Hz /1Ch/16 bit) are equally segmented into four blocks  $x_0, x_1, x_2$  and  $x_3$ , and

each block divided into  $p$  length sub-blocks is stored in a block buffer. A fifth block buffer is used to store the FIR impulse response of  $M$  coefficients.

The filtering stage is a parallel structured of 1-D FFT convolver array of the single FFT convolvers as shown in Figure 3.10. The FFTs N-point size must be long enough that circular convolution does not take place. This means that the N-point FFT should be the same length as the output segment,  $N=p+M-1$ . For instance, if the filter kernel contains 255 coefficients and each input segment contains 258 points, making output segment  $(255+258-1= 512)$  points long. This calls for 512-point FFTs to be used. That means the filter kernel must be padded with  $(512-255=257)$  zeros to bring it to a total length of 512 points. Likewise, each of input segments must be padded with  $(512-258=254)$  zeros. Each block of  $x(n)$  and  $h(m)$  padded with  $M-1$  zeros and  $p-1$  zeros respectively. After zero padding, the FIR impulse response coefficient and the speech signal blocks are simultaneously frequency transformed via the real inputs of five parallel Xilinx FFT v7\_0 blocks, and setting the imaginary input to zero. Each FFT outputs a frequency spectrum of two parts, real and imaginary. Subsequently, a parallel point-by-point complex multiplication is performed.

The Inverse FFT is then synchronized by the data available (dv) output of the FFT block to find the sub-output segment from its frequency spectrum. The real and imaginary parts of the multiplication result bit streams are bit manipulated first, then, fed to four Xilinx Inverse FFT v7\_0 blocks; so that each filtered sub-speech sequence can be transformed back to the time domain. The input frequency signals must be first scaled by  $\left(\frac{1}{N}\right)$  using Shift block to perform  $(\log_2 N)$  bit right shift. Then, the bit growth within the resultant scaled frequency sequences is converted back to the normalized 16-bit word-length with a binary point inserted at the 15-bit by a Convert block.

The output stage is a parallel to serial pipelined process of reconstruction, recording and displaying filtered output. The reconstruction is achieved using three pipelined adder trees. The pipelining of the adder tree is sized according to the overlap-add technique, as shown in Figure 3.12. The final output is connected to a gateway-out block to convert the fixed point format to a floating point for recording as a WAV file and displaying.



**Figure 3.12: Architecture 5; implementation of Fast Parallel 1-D FFT block Filtering Algorithm in the Virtex-6 FPGA board**

### 3.5.3 Performance Indices of Fast Parallel 1-D FFT Filtering Architecture

Fast 1-D FFT filtering architecture results are presented as area occupation and performance indices of the two architectures are indicated in Table 3.4 and Table 3.5, while the filtered outputs depicted in Figure 3.13 using architecture 5. For the same input signal, the two main variables that have affected the results are the FIR filter kernel length and the size of the N-point FFT/IFFT. Hence, comparative evaluation results for two FPGA implementation architectures are obtained for a distinctive linear phase FIR impulse response at lengths 3, 7, 15, 31, 61, 127, 255, 511, 1023 and 2047 coefficients, to be individually applied to an 8, 16, 32, 64, 128, 256, 512, 1024, 2048 and 4096 N-point FFT/IFFT, respectively. Area wise, architecture 4 and 5 logic area occupations are summarized in Table 3.4, which shows that architecture 4 was occupying, in average, less logic devices of (37%) FFs, (40%) LUTs, (339%) slices, (14%) DSPs and, in average, (34%) block RAMs than architecture 5. Due to that architecture 4 is a single convolver unit implementation compared to the parallel structure of the 1-D FFT convolution units in architecture 5.



Several observations can be made from the results in Table 3.5, which presents three performance indices of frequency, dynamic power and throughput. Firstly, the frequency and power was improved using architecture 4, compared to those of architecture 5 within all the  $(h(m), N\text{-point FFT})$  pairs. Secondly, architecture 5's throughput outperformed that of architecture 4, in average, by four-fold within all the  $(h(m), N\text{-point FFT})$  pairs. Thirdly, the first performance index, in Table 3.5, is maximum clock frequency are steadily rising up, for both architectures, as the  $(h(m), N\text{-point FFT})$  pairs descending. Fourthly, the power consumption is monotonically decreases, for both architectures, as the  $(h(n), N\text{-point FFT})$  pairs descending.

**Table 3.4: Logic Devices utilization by the parallel 1-D FFT filtering' architectures**

FFT N-point	Architecture 4					Architecture 5				
	FFs	LUTs	Slices	DSP 48E1s	RAMB 18E1s	FFs	LUTs	Slices	DSP 48E1s	RAMB 18E1s
8	3407	2010	677	18	8	9166	4852	1931	107	23
16	4420	2736	1002	18	8	11641	6226	2742	125	23
32	6258	3961	1316	27	8	16645	9628	3916	170	23
64	7502	4842	1746	27	8	20161	11430	4575	188	23
128	9586	6440	1970	36	8	25903	15762	6148	251	23
256	11036	8062	2374	36	5	29581	19128	7190	269	23
512	13036	10603	3422	52	5	35293	24975	8895	367	23
1024	14327	11273	3296	52	14	39245	28021	9844	385	37
2048	17182	11976	3563	70	32	47286	32365	11642	475	87
4096	18912	14366	4137	74	66	51899	35874	12491	511	174

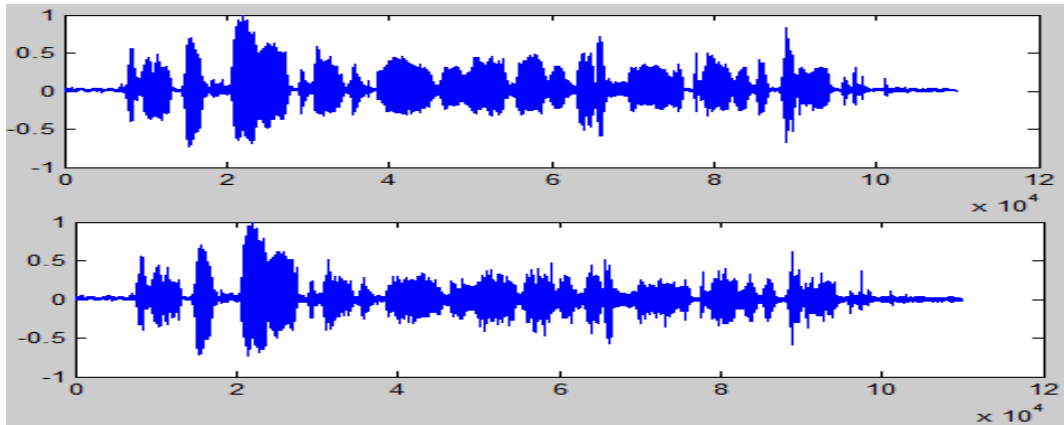
**Table 3.5: Performance indices of the parallel 1-D FFT filtering' architectures**

FIR Filter impulse response (M)	FFT N- point Size (N)	Maximum clock frequency (MHz)		Dynamic Power (mWatt)		Throughput (BPS)	
		Archit. 4	Archit. 5	Archit. 4	Archit. 5	Archit. 4	Archit. 5
3	8	277	243	15	36	150	599
7	16	248	225	19	45	135	555
15	32	239	211	23	69	130	520
31	64	226	201	29	78	123	495
63	128	217	193	31	88	118	476
127	256	208	184	37	103	113	454
255	512	199	176	42	120	108	434
511	1024	187	165	43	127	101	407
1023	2048	165	152	58	172	88	375
2047	4096	154	139	78	233	83	343

For comparison purposes, a parallel 1-D block filter algorithm based on the overlap-add approach was implemented on multi-DSPs platform in the ASP-PI5 DSP card [25]. An input of length  $(N = 4035)$  and a variable length impulse response filter  $(m = 15, 31, 61)$  were used; FIR filtering was carried out using the complex Fast Fourier transform (FFT) transform provided by the DSP library of functions. 1-D filtering results were obtained using single DSP processor and parallel 4-DSP system. The 4-parallel DSP system

achieved a high speed up factor, close to the number of processors used. The performance indices of logic area, power consumption and throughput were not taking into consideration.

The linear phase FIR filter kernels, as defined in subsection 3.2.2, are designed greater than 60 up to 2047 coefficients using the Xilinx FDATool block. A sample of the filtering results is shown in Figure 3.13 using architecture 5 at  $M= 63/ N=128$ . The filtered speech signals are enhanced and slightly longer by  $M-1$  points.



**Figure 3.13: Architecture 5's filtering results of the noisy real-time speech (22050 Hz/ 1Ch/ 16 bit) signal at  $M= 63/ N=128$**

### 3.6 1-D Cross-Correlation Application: FPGA Architecture for Parallel 1-D Matched Filtering Algorithm

Detection of a known waveform in a noisy signal is the fundamental problem in signal analysis of, including but not limited to, medical acoustic sound [81] and echo location systems[93]. Correlation is the optimal technique for detecting a known waveform in a noisy signal which called matched filtering. That is, the peak is higher above the noise using correlation than can be produced by any other linear system. The filter kernel of the matched filter is the target signal being detected, except it has been flipped left-for-right. This flip is required to perform correlation using convolution, making it extremely slow to execute. Then, speed improvement can be implemented using parallel FFT convolution of architecture 5 to assure high throughput.

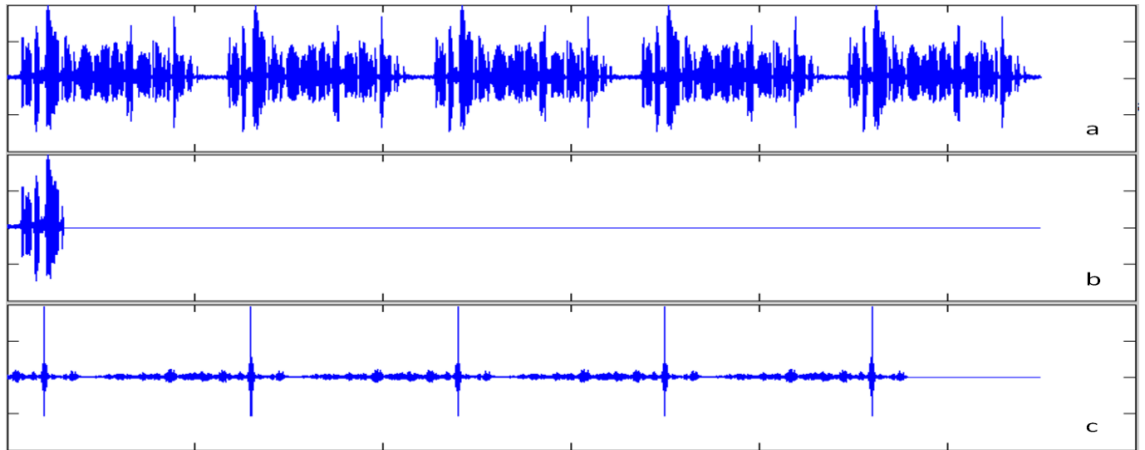
Therefore, architecture 5 can be effectively modified for parallel matched filtering to detect a known segment  $x(m)$  within the 1-D noisy input signal  $x(n)$ . Where, the FIR impulse response coefficients of the filter are replaced by the samples of the target input segment to be detected. Except, the target segment  $x(m)$  has been flipped left-for-right to implement its reversed-time version  $x(-m)$ . For example, when  $x(n)$  and  $x(m)$ , are convolved to produce  $y(u)$ , the equation will be as:

$$y(u) = \sum_{r=0}^{k-1} \sum_{n=0}^{p-1} x_r(n + rp) x(n - m) \quad (3.17)$$

In comparison, the correlation  $z(u)$  of  $x(n)$  and  $x(m)$  can be obtained by convolution as:

$$z(u) = \sum_{r=0}^{k-1} \sum_{n=0}^{p-1} x_r(n + rp) x(n - (-m)) \quad (3.18)$$

The matched filter output signal, as shown in Figure 3.14 has a far higher peak than the residue noise in the input signal, provided by any other linear filter. The amplitude of each point in the output signal is a measure of how well the filter kernel matches the corresponding section of the input segment. Consequently, the output of a matched filter does not necessarily look like the signal being detected.



**Figure 3.14: The matched filtering results using architecture 5 at  $M= 63/ N=128$ .  
(a) Input speech signal, (b) Target input segment, (c) Matching results.**

### 3.7 Conclusion

A generic parallel hardware versions of the parallel 1-D convolution-filtering algorithm were developed for the multi-MAC FIR and FFT convolution, to cover the entire range of linear FIR filter length of 3, 7, 15, 31, 61, 127, 255, 511, 1023, 2047, ... coefficients. The filtering methods were based on the overlap-add block filtering. Then, FPGA-based implementations of five generic architectures on Virtex-6 ML605 board were developed. The performance indices for the five architectures were evaluated as a complete package of area, speed, power and throughput using XSG to target a Virtex-6 ML605 board. A practical example was developed, implemented and analysed to realize cross-correlation as a parallel 1-D matched filter algorithm for real-time speech signature detection using architecture 5.

## Chapter 4. **Parallel 2-D Greyscale/Colour Image Filtering Algorithm and Its FPGA Implementations**

### **4.1 Introduction**

Digital image processing frequently exploits convolution and correlation functions to achieve linear 2-D image filtering algorithmic applications, for example, in mobile phones [94], consumer electronics such as digital cameras [95], medical imaging [21, 35, 96, 97], brain-computer interface [98], computer vision [99] and satellite images enhancement [100]. The demands for linear 2-D image filtering have inevitably increased in these recent applications for noise reduction, edge extraction, similarity detection and enhancement [35, 97, 101]. However, most of the digital image filtering systems have an inherently bottlenecked when intensive processing of the large 2-D images input with the extensive range of the linear 2-D image filtering kernels is required, due to the lack of memory capacity, data communication overhead and intensive computational power [15, 32], all of which slows down these filtering algorithms.

To resolve these bottlenecks, the large 2-D input data is decimated into independent sub-data input that can be simultaneously processed in parallel. Consequently, a parallel 2-D image filtering algorithm is proposed which decimates the large 2-D image, independently filtering in parallel, without the need for the communication among processing stages, then interpolating to reconstruct the filtered image.

To automate [34, 35, 102-107] the proposed parallel 2-D image filtering algorithm, the FPGA [101, 102, 105, 108] is exploited for its intrinsic parallelism of logic area and IP cores to achieve the required level of performance for many digital image processing applications. The main concern with the FPGA is the overall performance indices of area, speed, dynamic power consumption and throughput [16-18]. Thus, a new generic parallel 2-D filtering algorithm is presented, mathematically modelled, implemented for both spatiotemporal and spectral architectures and their complete performance indices package are evaluated.

This chapter is organised in eight sections. In section 4.2, the main research concepts are defined and mathematically explained. In section 4.3, the generalized parallel 2-D linear image filtering algorithm is presented, and the mathematically modelled. In section 4.4, seven parallel spatiotemporal filtering architectures are developed. Then, their performance indices are tabulated, analysed and discussed as a complete set of area,

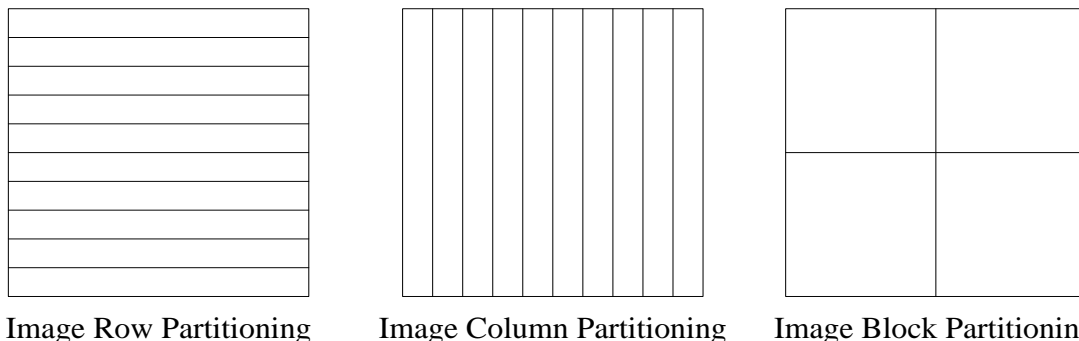
speed, dynamic power consumption and throughput as well as the computation rate. In section 4.5, a successful application of 3-D colour MRI slice filtering using the seven generic architectures is developed. In section 4.6, three FPGA implementations as parallel generic FFT architectures are developed and their performance indices are evaluated as a complete set of area, speed, dynamic power consumption and throughput. In section 4.7, the implementation of a 2-D cross-correlation function is realized for parallel 2-D MRI match filtering algorithm. Finally, the conclusion of this chapter is presented in section 4.8.

## 4.2 Research Concepts Definitions

This section introduces the main research concepts on which this chapter is based.

### 4.2.1 Linear 2-D Stream Filtering Method

One of the common bottlenecks within image filtering is the time and memory bandwidth required to read the image from memory and write the filtered image to memory. Stream filtering [103, 109] can overcome this bottleneck in two steps of parallelism [34, 107]. Firstly, spatial parallelism may be exploited by splitting the image into blocks of rows, columns or squares, as illustrated in Figure 4.1.



**Figure 4.1: A single digital image is partitioned into spatial parallelism of pixels for parallel digital processing.**

Secondly, convert spatially partitioned blocks into temporal parallelism, as illustrated in Figure 4.2. The image is often streamed at a rate of one pixel per clock cycle. The parallel filtering operations are performed on-the-fly for the pixels using an independent 2-D FIR filters on each partition block. Architectures 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15 are developed to achieve the linear image filtering based on these two levels of parallelism. Thus, the linear 2-D stream filtering method exploits, as much processing should be performed on the streamed pixels as passing through the FPGA.



$$\sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} h(m_1, m_2) = \begin{cases} 0 \\ 1 \end{cases} \quad (4.3)$$

The coefficients sum is zero for the edge detection filters, or one for noise smoothing filters. So that, the edge filtering response is zero for the regions of uniform pixel value, While, the noise smoothing output values are not different from the input in uniform regions of the image. The edge detection can be enhanced by summing weights up to one, therefore, the edge filter is called edge enhancement filter.

Some of the common generic edge filters are Edge, Sobel, Laplacian and Prewitt, and for the noise smoothing filters, Sharpen, Gaussian, Smooth and Blur, as shown in Table 4.1 for the (5×5) operators. These 2-D filters will be implemented and applied to the biomedical imaging of Magnetic Resonant Imaging (MRI) as a “plug and filter” operator using the developed FPGA-based architectures. This 2-D image processor can be realized in the spatiotemporal domain, as in architectures 6, 7, 8, 9, 10, 11 and 12 or the frequency domain as in architecture 13, 14 and 15.

**Table 4.1: some of the common edge and noise smoothing FIR filter of 5x5 kernels**

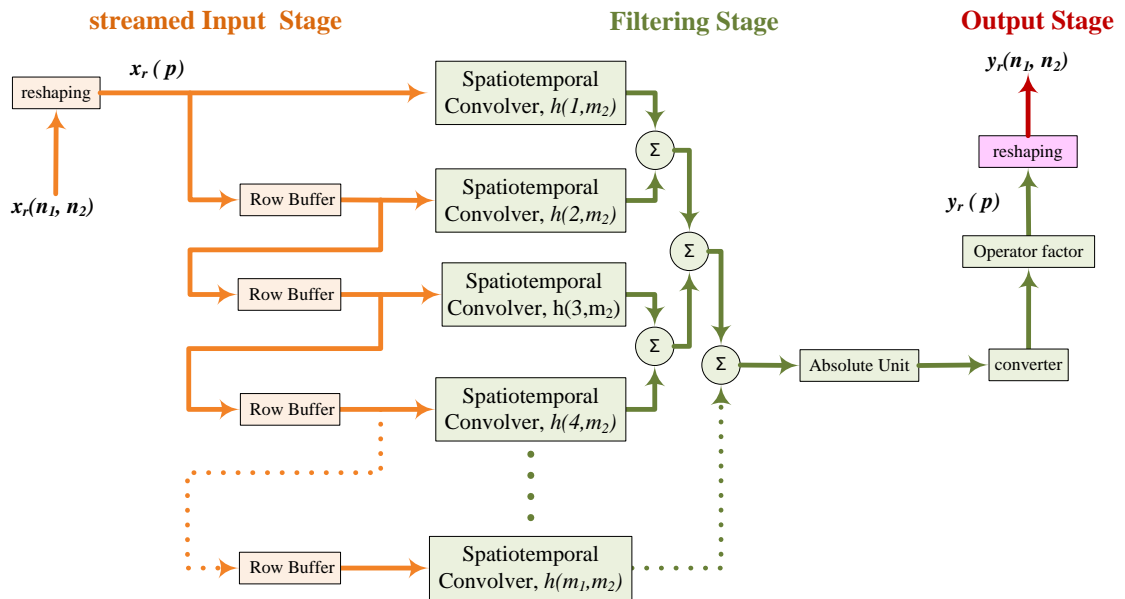
2-D FIR Filter	5x5 kernel	2-D FIR Filter	5x5 kernel
<b>EdgeXY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & -1 & 8 & -1 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	<b>Laplacian</b> D.F= $(\frac{1}{8})$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & -8 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
<b>EdgeX</b>	$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$	<b>PrewittX</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
<b>EdgeY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	<b>Blur</b> D.F= $(\frac{1}{16})$	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$
<b>SobelXY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	<b>Smooth</b> D.F= $(\frac{1}{100})$	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 5 & 44 & 5 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$
<b>SobelX</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -2 & 0 & 2 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	<b>Sharpen</b> D.F= $(\frac{1}{16})$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & -2 & -2 & 0 \\ 0 & -2 & 32 & -2 & 0 \\ 0 & -2 & -2 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
<b>SobelY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	<b>Gaussian</b> D.F= $(\frac{1}{52})$	$\begin{bmatrix} 1 & 1 & 2 & 1 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 2 & 4 & 8 & 4 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 1 & 1 & 2 & 1 & 1 \end{bmatrix}$

The filter kernel size ( $M_1 \times M_2$ ) determines the proper convolution implementation for the linear filter to be in the spatiotemporal or frequency [17, 82, 110]. Spatiotemporal or direct convolution is more important of these two, since images have their information encoded in the spatiotemporal domain rather than the frequency domain. However,

direct convolution has an execution time proportional to  $(N_1 \times N_2 \times M_1 \times M_2)$  for an  $(N_1 \times N_2)$  image convolved with an  $(M_1 \times M_2)$  kernel. Consequently, the execution time for direct convolution depends very strongly on the size of the kernel used for a particular input image. While, the frequency or a radix-2 FFT convolution is effectively filtering a particular image in  $(N_1 N_2 \log_2 N_1 N_2)$  steps. By comparison,  $(M_1 \times M_2)$  is always greater than  $(\log_2 N_1 N_2)$  for a filter kernel of thousands of coefficients [17, 82]. Consequently, the 2-D linear convolution of (4.2) can be efficiently implemented directly in the time domain, or from their effects in the frequency domain. For filter kernels shorter than about  $10 \times 10$ , the spatiotemporal convolution is implemented as  $2 \times 2$ ,  $3 \times 3$ ,  $5 \times 5$ ,  $2 \times 4$ ,  $4 \times 4$ ,  $2 \times 8$  and  $8 \times 8$  kernels. While, the FFT convolution is implemented for the longer filter kernels of  $15 \times 15$ ,  $31 \times 31$ ,  $63 \times 63$ ,  $127 \times 127$  and  $255 \times 255$ .

### 4.2.3 Spatiotemporal Convolver Engine

The generic 2-D spatiotemporal convolver engine is a  $(M_1 \times M_2)$  MAC FIR digital filter. That consists of three stages: spatial to temporal parallelism,  $M_1$  spatiotemporal convolver units and sub-filtered merging stages, as shown in Figure 4.3.



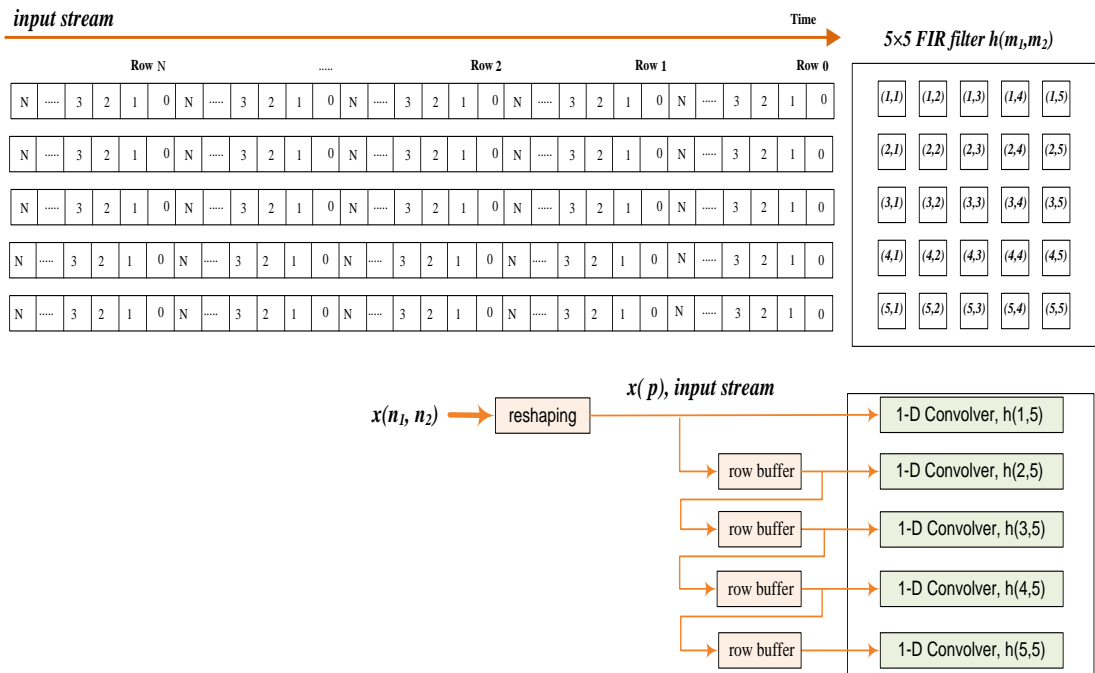
**Figure 4.3: Implementation of the Generic 2-D Spatiotemporal Convolver Unit**

The spatial partitioned block of a digital image is sequentially streamed into  $(M_1 - 1)$  row buffers to be filtered in parallel using the  $M_1$  spatiotemporal convolver units. Each row buffer effectively delays the input by one row of  $N_1$  pixels, where  $N_1$  is the image width. An adder tree merges the sub-filtered streams. The resulting pixel values, after applying the 2-D image filter, can be negative or larger than 255. Thus, the resulting pixel values



are conditioned by taking the absolute value from the negative results, restrict the bit growth to 8 bits and truncate pixel values larger than 255 to 255 by streaming into an absolute unit and Xilinx convert block respectively. The conditioned pixels are multiplied by a constant ratio factor as per the 2-D operator matrix. For example, as shown in Table 4.1, the sharper operator have a normalization factor of  $(1/16)$  to ensure that the low spatial frequencies are not amplified.

This linear image filtering considers two inputs of a 2-D filter matrix, and a 2-D image matrix. Where, the two matrices move over every element of the other matrix. Logically, there are two implementation strategies: either by scanning, where the 2-D filter matrix scans the 2-D image matrix, or by streaming, where the 2-D image matrix streams into the 2-D filter matrix. For example, consider a  $5 \times 5$  window filter, each output filter is a function of the twenty-five pixel values within the window. Without stream filtering, twenty-five pixels must be read, a pixel at each clock cycle, for each window position and each pixel must be read twenty five times as the window is scanned through the image.



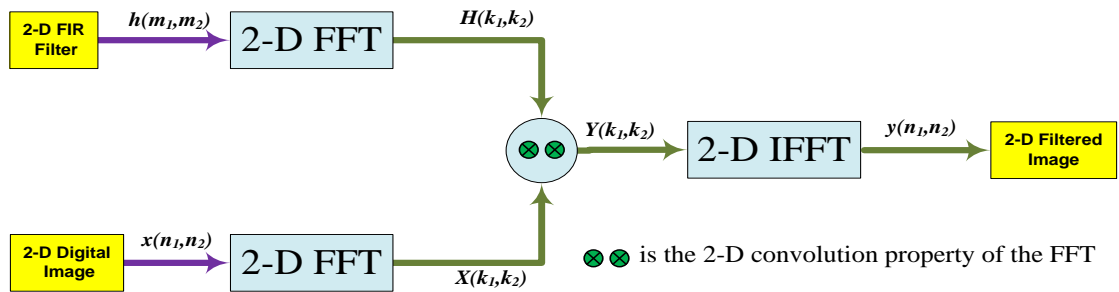
**Figure 4.4: Five temporal parallelism copies of a digital image are streamed into a  $5 \times 5$  FIR filter**

A  $5 \times 5$  filter kernel spans five image rows, the current row and four previous rows. Alternatively, the stream filtering stores the  $5 \times 5$  filter kernel in a spatiotemporal convolution engine, and streams five temporal parallel copies of the digital image into five multi-MAC convolvers. The first temporal pixel copy is streamed into the first FIR

engine of  $h(1,5)$  row of the 2-D operator matrix. Each next temporal pixels copy is delayed by one row of  $N_l$  pixels, then, streamed into the next row of the 2-D operator matrix, as shown in Figure 4.4. The spatiotemporal convolver are of three parallel types single, dual and quad MAC, as will be explained in subsection 4.4.

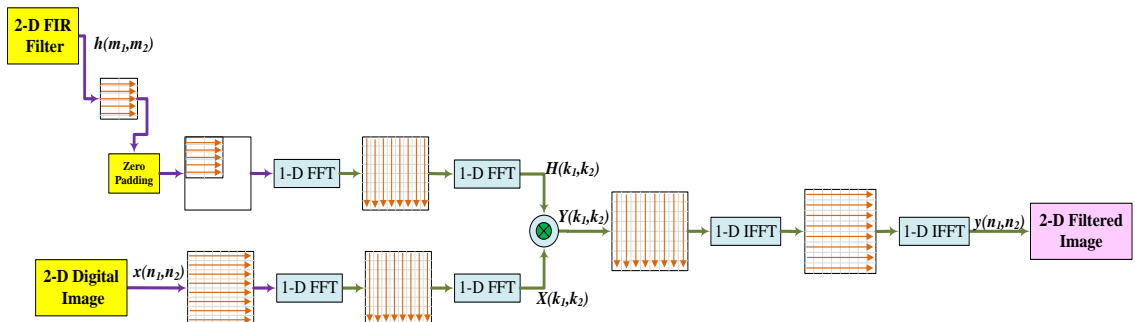
#### 4.2.4 2-D FFT Convolver Engine

2-D Convolution in the time domain, as stated in (4.2), corresponds to complex multiplication of 2-D FIR filter and digital 2-D image spectra, according to the circular convolution property of the FFT [16, 17, 82], as shown in Figure 4.5. Multiplication costs less logic area in digital implementation than convolution. This Fourier image analysis property[17, 46, 82], then, enables 2-D FFT convolution to be utilized in linear image filtering.



**Figure 4.5: 2-D FFT convolver unit**

The 2-D FFT in Figure 4.5 calculates the Fourier transform of a 2-D FIR filter and a 2-D digital image. Since the Fourier transform is inherently separable [17, 87], then the 2-D FFT can be reduced to a 1-D FFT pair, which is called row-column FFT [33]. Therefore, the 2-D FFT can be calculated by taking the 1-D FFT of each row, followed by the 1-D FFT of each column, as shown in Figure 4.6. Since, the Fourier transform is



**Figure 4.6: 2-D FFT convolver unit is separable to be implemented as 1-D FFT pair engine architecture.**

complex value in the frequency domain, the image will be transformed into an intermediate image of real array and imaginary array. Next, 1-D FFT is repeated on

each column of the intermediate image. The resulting real and imaginary parts are the image's 2-D frequency spectrum,  $X(k_1, k_2)$ .

Similarly, 1-D FFT pair calculates the 2-D FIR filter's 2-D frequency spectrum,  $H(k_1, k_2)$ . These two frequency spectra are point-by-point multiplied to produce  $Y(k_1, k_2)$ , which represents the 2-D FFT of the filtered image. To transform back into the time domain as  $y(n_1, n_2)$ , the inverse Fourier transform of  $Y(k_1, k_2)$  is calculated by taking the 1-D inverse FFT of each column, followed by the 1-D inverse FFT of each row.

The number of complex multiplications and additions required for a radix-2 1-D FFT algorithm of length  $N_1$  is  $(\frac{N_1}{2}) \log_2 N_1$  and  $N_1 \log_2 N_1$ , respectively. Thus, the number of complex multiplications and additions needed for the row-column FFT that employs such a 1-D FFT are  $(\frac{N_1 N_2}{2}) \log_2 N_1 N_2$  and  $N_1 N_2 \log_2 N_1 N_2$ , respectively. Therefore, the row-column FFT reduces the computational complexity from  $O(N^4)$  to  $O(N^2 \log_2 N^2)$ . This considerable gain in computation justifies the hardware implementation of the row-column FFT over the 2-D DFT [55, 111].

#### 4.2.5 Total 2-D Throughput

The total throughput  $\mu$  of the parallel 2-D MAC FIR filtering architecture is measured by frame per second (FPS). Generally, the maximum throughput is the maximum operating clock frequency divided by one frame [20, 79, 80] at a filtering rate of one pixel operation per clock cycle. However, the throughput of the parallel 2-D MAC FIR filtering architecture is limited by the large input image and 2-D MAC FIR operation, which can be mitigated by parallelism. Thus, the total throughput is directly proportional to the operating frequency and the levels of parallelism, and inversely proportional to the size of the input image and the 2-D MAC FIR matrix.

The levels of parallelism is the 2-D image input decimation by  $\alpha_1$  and the number of multi-MAC engines  $\alpha_2$  per convolver unit. Then, the total throughput  $\mu$  can be formulated as:

$$\mu = \frac{\mu_{FIR}}{N_1 \times N_2} \times \alpha_1 \quad (4.4)$$

Where,  $N_1 \times N_2$  is the input image frame dimensions.

The 2-D MAC FIR engine throughput  $\mu_{FIR}$  is directly proportional to the clock speed  $f$  and limited by the 2-D FIR coefficients ( $M_1 \times M_2$ ) which can be distributed to be processed by multi-MAC engines. Then  $\mu_{FIR}$  can be stated as:

$$\mu_{FIR} = \left( \frac{f}{M_1 \times M_2} \right) \times \alpha_2 \quad (4.5)$$

Thus, the total throughput for the parallel 2-D MAC FIR filtering architectures can be calculated as:

$$\mu = \frac{f \alpha_1 \alpha_2}{N_1 N_2 M_1 M_2} \quad (4.6)$$

where,  $\frac{N_1 N_2}{\alpha_1} \frac{M_1 M_2}{\alpha_2}$  is equivalent to one frame at a filtering rate of one pixel operation per clock cycle.  $\alpha_1 = 2 \times 2$ ,  $\alpha_2 = 1, 2$  or  $4$  depending on the degree of parallelism inside the 2-D spatiotemporal convolver using single MAC, dual MAC or quad MAC engine.

For the parallel 2-D FFT filtering architectures, one frame time is  $\frac{N_1 N_2}{\alpha_1} \log_2 \frac{N_1 N_2}{\alpha_1}$  because the single 2-D FFT unit performs the 2-D convolution in  $O\left(\frac{N_1 N_2}{\alpha_1} \log_2 \frac{N_1 N_2}{\alpha_1}\right)$  steps [44], while the spatiotemporal convolution requires  $O\left(\frac{N_1 N_2}{\alpha_1} \frac{M_1 M_2}{\alpha_2}\right)$  steps.

Thus, the overall throughput  $\mu$  of the parallel 2-D FFT convolution architecture for 2-D image input decimated by 2 and maximum clock frequency  $f$  (MHz) can be formulated as:

$$\mu = \frac{f}{\frac{N_1 N_2}{\alpha_1} \log_2 \frac{N_1 N_2}{\alpha_1}} \quad (4.7)$$

The throughput for the parallel 2-D convolution architectures depends only on the image size for the FFT convolution, while spatiotemporal convolution depends on both the image and the kernel size.

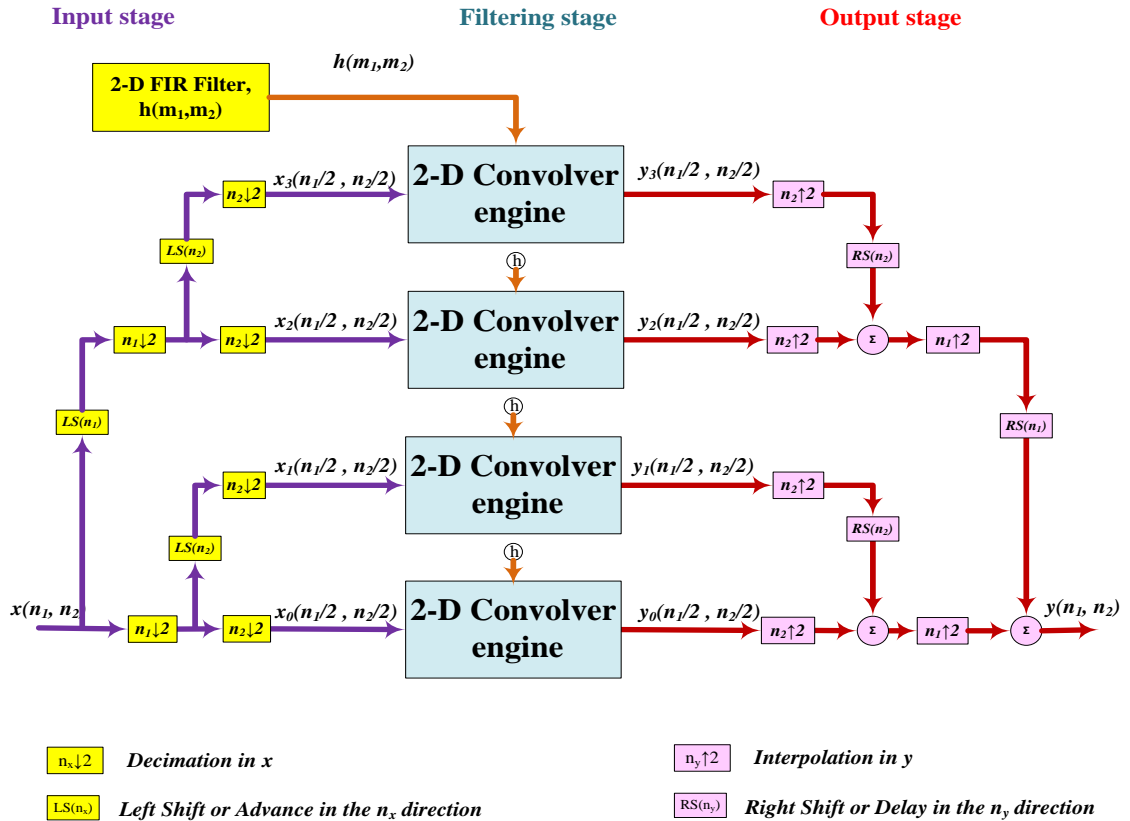
#### **4.2.6 Total Computation Rate**

The total number of MACPS can be considered as another performance index to indicate the computation rate for each one of the seven spatiotemporal convolution architectures developed. The total computation rate  $\gamma$ , measured in GMACPS, is directly proportional to the levels of parallelism and the maximum clock frequency  $f$ . The level of parallelism is the number of parallel 2-D spatiotemporal convolver stages ( $\alpha_1$ ) and the number of MAC engines ( $\alpha_2$ ) per multi-MAC and the number of multi-MAC convolvers  $\alpha_3$  per 2-D spatiotemporal convolver. Thus, the computation rate can be formulated as;

$$\gamma = f \times \alpha_1 \times \alpha_2 \times \alpha_3 \quad (4.8)$$

### 4.3 The Generalized Parallel 2-D Linear Image Filtering Algorithm

The generalized parallel 2-D filtering algorithm, as shown in Figure 4.7, for the linear image stream filtering method is implemented for both the spatiotemporal and frequency convolvers. The spatiotemporal convolver is developed in the FPGA as architectures 6, 7, 8, 9, 10, 11 and 12. In addition to, the frequency or 2-D FFT convolver is realized as architectures 13, 14 and 15. This parallel filtering algorithm consists of three stages: input decimation stage, parallel sub-filtering processing stage and parallel to serial interpolation output stage. The three stages are implemented on the Virtex-6 ML605 FPGA board using XSG. The mathematical model of these three stages for the stream filtering method are presented in the following subsections.



**Figure 4.7: The Generalized Parallel 2-D Linear Image Filtering Algorithm**

#### 4.3.1 Input Decimation by 2 for the 2-D Image Stream Filtering

The input grayscale image  $x(n_1, n_2)$  of size  $(N_1 \times N_2)$  is decimated by 2 in the two dimensions producing  $(2^2=4)$  sub-image blocks of size  $(\frac{N_1}{2}, \frac{N_2}{2})$  for each 2-D image input, as shown in Figure 4.7. The resultant 2-D sub-image blocks are defined as:

$$\begin{aligned}
x_0 &= x(2n_1, 2n_2) \\
x_1 &= x(2n_1, 2n_2+1) \\
x_2 &= x(2n_1+1, 2n_2) \\
x_3 &= x(2n_1+1, 2n_2+1)
\end{aligned} \tag{4.9}$$

The decimated blocks of the input 2-D image are realized by a left shift and decimation in two dimensions, which are indicated in Figure 4.7 by  $LS(n_1)/LS(n_2)$  and  $n_1 \downarrow 2 / n_2 \downarrow 2$  respectively. These four 2-D sub-images are not overlapped, and have exactly the same pixels as the original 2-D image, hence, the decimation process did not lose or reduce any pixel of the original 2-D image. Thus, these four 2-D sub-images can be filtered simultaneously and independently within the 2-D processing stage.

### 4.3.2 Parallel 2-D Filtering Stage

The 2-D processing is achieved by convolving the sub-images given by (4.10) with the 2-D FIR kernel  $h(m_1, m_2)$ . This linear 2-D filtering can be realized by either spatiotemporal or FFT convolution, and expressed by:

$$y_r \left( \frac{n_1}{2}, \frac{n_2}{2} \right) = x_r \left( \frac{n_1}{2}, \frac{n_2}{2} \right) ** h(m_1, m_2), \quad r=0,1,\dots,3 \tag{4.10}$$

where,  $**$  is the 2-D convolution operation. Equation (4.10) describes the filtering process shown in Figure 4.7, which is computationally intensive. Thus, a parallel array of independent 2-D convolution units can be used to produce a filtered 2-D image, speeding up the filtering rate, increasing the throughput and carrying out real time processing of large input image sizes. That can be detailed by the following parallel linear convolutions:

$$y_r(n_1, n_2) = \sum_{r=0}^3 \left( \sum_{m_1=0}^{\frac{N_1}{2}-1} \sum_{m_2=0}^{\frac{N_2}{2}-1} x_r(m_1, m_2) h(n_1 - m_1, n_2 - m_2) \right) \quad r = 0,1,\dots,3 \tag{4.11}$$

Where,  $0 \leq (n_1) < \frac{N_1}{2} + \frac{M_1}{2} - 1$  and  $0 \leq (n_2) < \frac{N_2}{2} + \frac{M_2}{2} - 1$  are the output dimensions of each sub-image.

### 4.3.3 Output Interpolation by 2 Reconstruction Stage

Four filtered streams of 2-D sub-image are produced by the parallel filtering stage. The resultant filtered outputs,  $y_0, y_1, y_2$  and  $y_3$ , are uniquely decimated by 2 sub-images as:

$$\begin{aligned}
y_0 &= y(2n_1, 2n_2) \\
y_1 &= y(2n_1, 2n_2 + 1, ) \\
y_2 &= y(2n_1 + 1, 2n_2, ) \\
y_3 &= y(2n_1 + 1, 2n_2 + 1)
\end{aligned} \tag{4.12}$$

The reconstruction of the final output  $y(n_1, n_2)$  is obtained by interpolations,  $n_1 \uparrow 2 / n_2 \uparrow 2$  in the two dimensions, and right shift of RS( $n_1$ )/RS( $n_2$ ) the filtered sub-images  $y_0, y_1, y_2$  and  $y_3$  given by (4.12) and can be seen in Figure 4.7. Therefore, the resultant filtered output can be reconstructed as shown below:

$$\begin{aligned}
y(1:2:n_1, 1:2:n_2) &= y_0 \\
y(1:2:n_1, 2:2:n_2) &= y_1 \\
y(2:2:n_1, 1:2:n_2) &= y_2 \\
x(2:2:n_1, 2:2:n_2) &= y_3
\end{aligned} \tag{4.13}$$

Where,  $j:i:k$  is the same as  $[j, j+i, j+2i, \dots, k]$ , and represents the two operations of right shift and interpolation by 2.

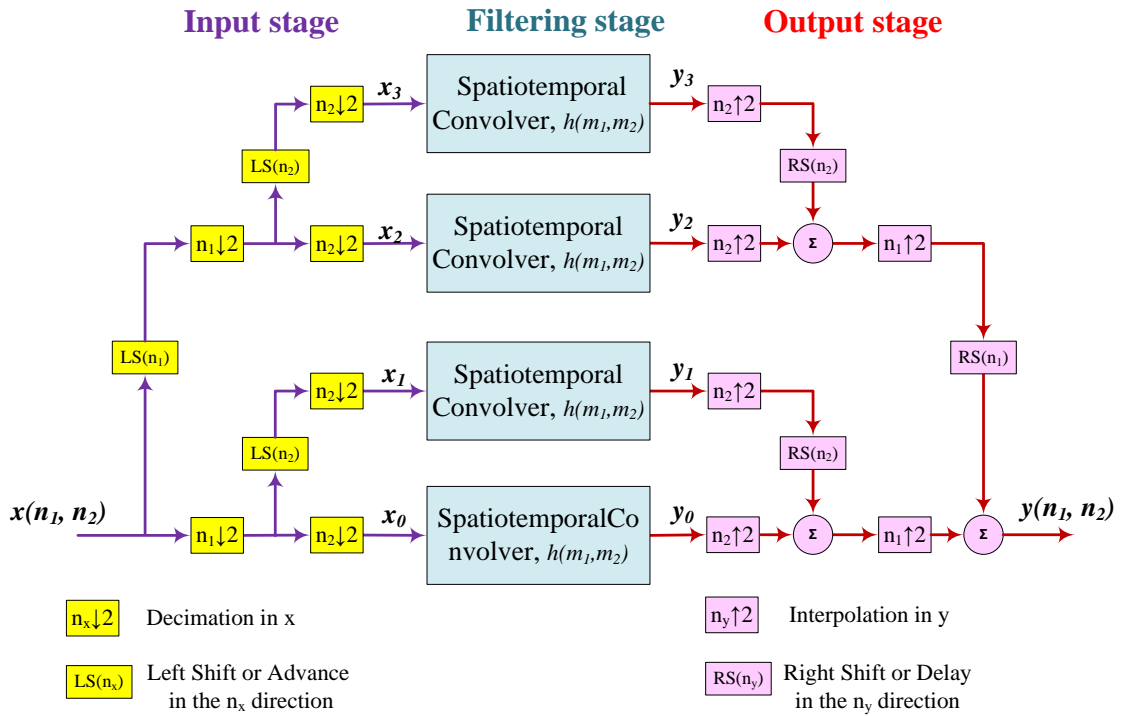
The generalized parallel 2-D linear stream-filtering algorithm can be realized in more than one unique architecture, as either a 2-D spatiotemporal convolution in the time domain, or a 2-D FFT convolution in the frequency domain, depending on the filter kernel length. Where, the parallel algorithm can directly be implemented faster by the MAC FIR convolution for the 2-D FIR kernel shorter than  $(10 \times 10)$  [82], the filtering time is proportional to the kernel length. However, longer filter kernels can efficiently be implemented faster with the 2-D FFT convolution, with very little penalty in filtering time. With 2-D FFT convolution, the filter kernel have no limits to exploit the appropriate size according to the application. The spatiotemporal and spectral image filtering is presented in subsection 4.4 and subsection 4.6 respectively.

#### 4.4 Parallel 2-D Spatiotemporal Convolver Architectures

The generalized parallel 2-D linear image filtering algorithm of Figure 4.7 can be efficiently realized by 2-D spatiotemporal convolution using multi-MAC FIR filter units as shown in Figure 4.8. Consequently, the 2-D image filtering applications [21, 35, 38, 109] in which the 2-D FIR kernel length shorter than  $(10 \times 10)$  coefficients can be achieved using the 2-D stream filtering as defined in sub-sections 4.2.1, 4.2.2 and 4.2.3. In the input stage, four spatially partitioned sub-image blocks, as shown in Figure 4.8,

of size  $(\frac{N_1}{2}, \frac{N_2}{2})$ . The parallel filtering operations are performed on-the-fly using an independent 2-D convolver engine for the pixel streams. The 2-D convolver engine is a  $(M_1 \times M_2)$  MAC FIR digital filter. The filter kernel of  $(M_1 \times M_2)$  coefficients are stored in the 2-D FIR filter, thus, there is no need for separate decimation.

The temporal partitioned digital image is sequentially streamed into  $(M_1 - 1)$  row buffers to be filtered in parallel using the  $M_1$  convolver units, this constitutes the 2-D  $(M_1 \times M_2)$  convolver engine. The spatiotemporal convolver architectures are of three parallelism type using single, dual and quad MAC units, as will be explained in subsection 4.4.1, 4.4.2 and 4.4.3 respectively. The reconstruction of the final output  $y(n_1, n_2)$  is obtained by interpolations,  $n_1 \uparrow 2 / n_2 \uparrow 2$ , and right shift of  $RS(n_1)/RS(n_2)$  the sub-images as described abstractly in (4.13) and shown graphically in Figure 4.8.



**Figure 4.8: Spatiotemporal implementation of the Parallel 2-D Convolution Algorithm**

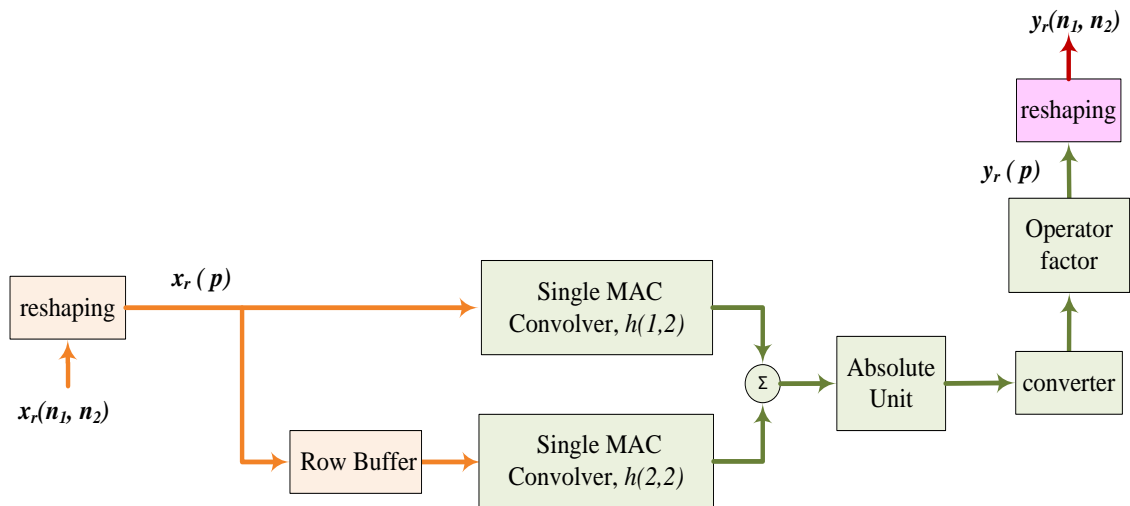
The parallel 2-D spatiotemporal convolution-filtering algorithm can be implemented by three parallel types of architectures according to the hardware structure of their 2-D convolver engine. The input decimated by 2 stage and the output interpolated by 2 stage are the same for all the seven architectures, the following architectures will be distinguished by their 2-D MAC engine structure. Thus, the general architecture, shown



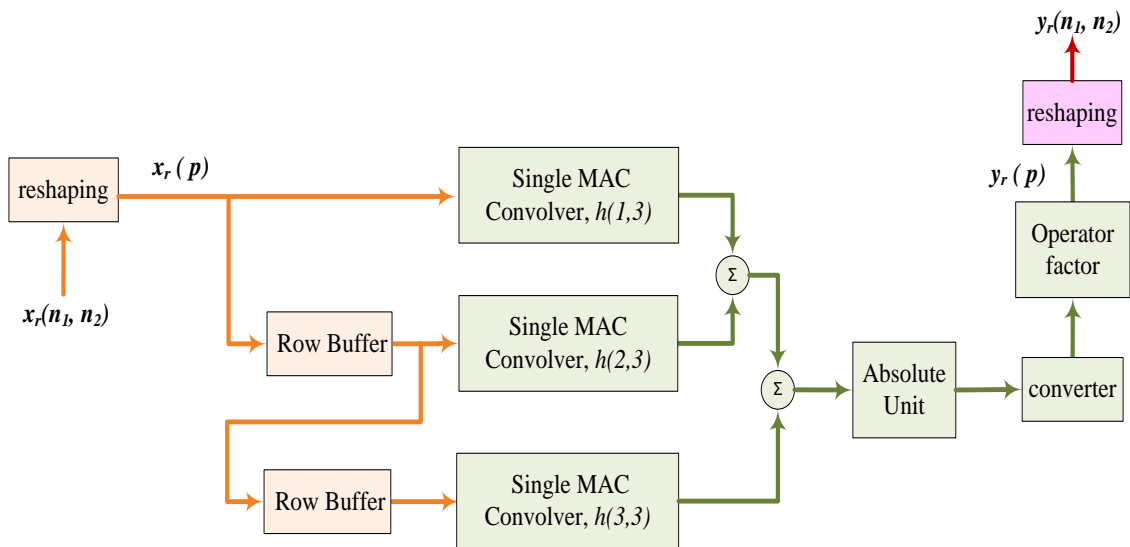
in Figure 4.8, can have more than one unique hardware version, as explained in the following subsections.

#### 4.4.1 Parallel 2-D Single MAC Convolver Architectures

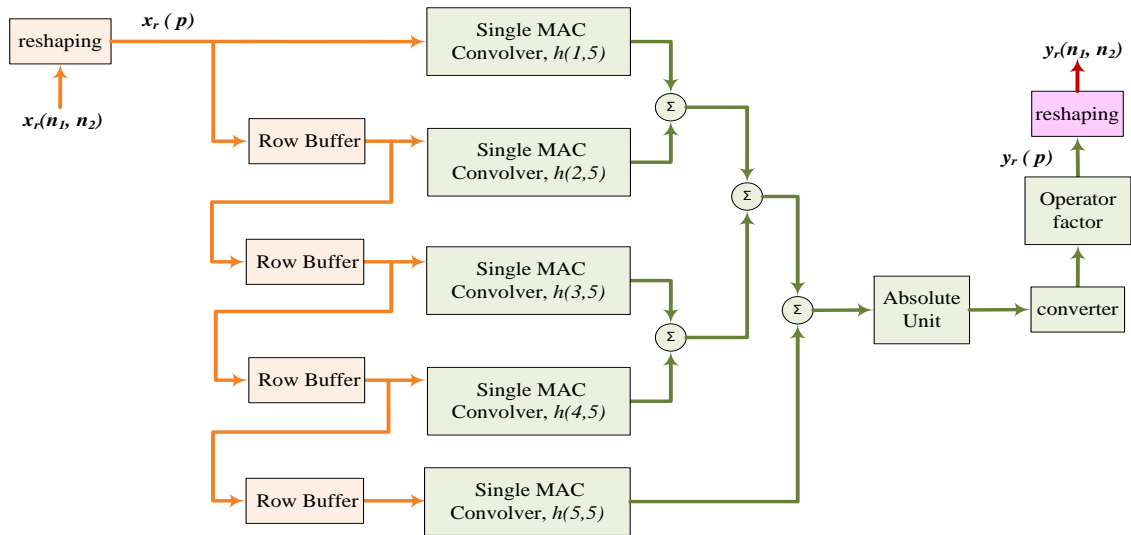
The spatiotemporal convolution architectures use 2-D single MAC convolver filter Units of size  $(2 \times 2)$ ,  $(3 \times 3)$  and  $(5 \times 5)$ , as architecture 6, 7 and 8 respectively. Figure 4.9, Figure 4.10 and Figure 4.11 show  $(2 \times 2)$ ,  $(3 \times 3)$  and  $(5 \times 5)$  single MAC 2-D direct convolver as unit 4, unit 5 and unit 6 respectively. These units realize the 2-D FIR operator of the filtering stage for the parallel linear image stream filtering algorithm of Figure 4.8. The row buffer can be implemented by a single Port RAM block with an up counter, as address generator, limited to the image width  $N_1$ , as shown in Figure 4.12.



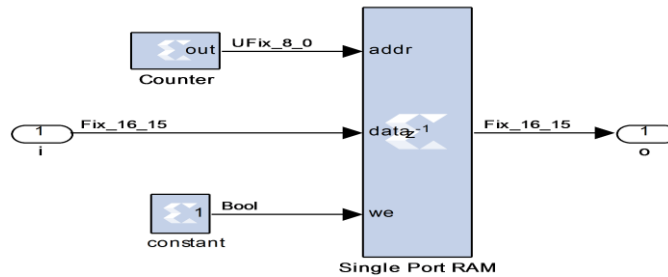
**Figure 4.9: Unit 4;  $(2 \times 2)$  Single MAC Convolver Architecture**



**Figure 4.10: Unit 5;  $(3 \times 3)$  Single MAC Convolver Architecture**



**Figure 4.11: Unit 6; (5x5) Single MAC Convolver Architecture**



**Figure 4.12: Row buffer implementation**

The Xilinx Slice block is parameterized to slice off the first top bit from the MSB of each input data sample. The output data type is unsigned one bit, either zero or one, with its binary point at zero. Thus, the multiplexer will select the d0 input for positive sample and d1 for negative samples after processing by a negate block.

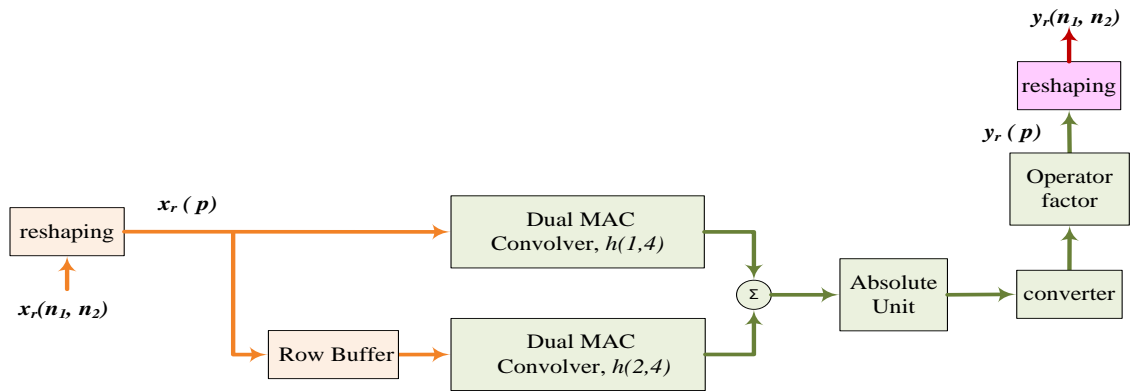
As configured and shown in Table 4.2, unit 4 occupies less logic area than unit 5 by (33%) FFs, (26%) LUTs, (50%) slices, (33%) dedicated DSP 48E1s multiplier, and (40%) RAMB 18E1s block memory. In addition, compared to unit 6, unit 4 occupies less logic area by (66%) FFs, (65%) LUTs, (68%) slices, (60%) dedicated DSP 48E1s multiplier, and (67%) RAMB 18E1s block memory.

**Table 4.2: Logic Devices utilization by the 2-D multi-MAC convolver units**

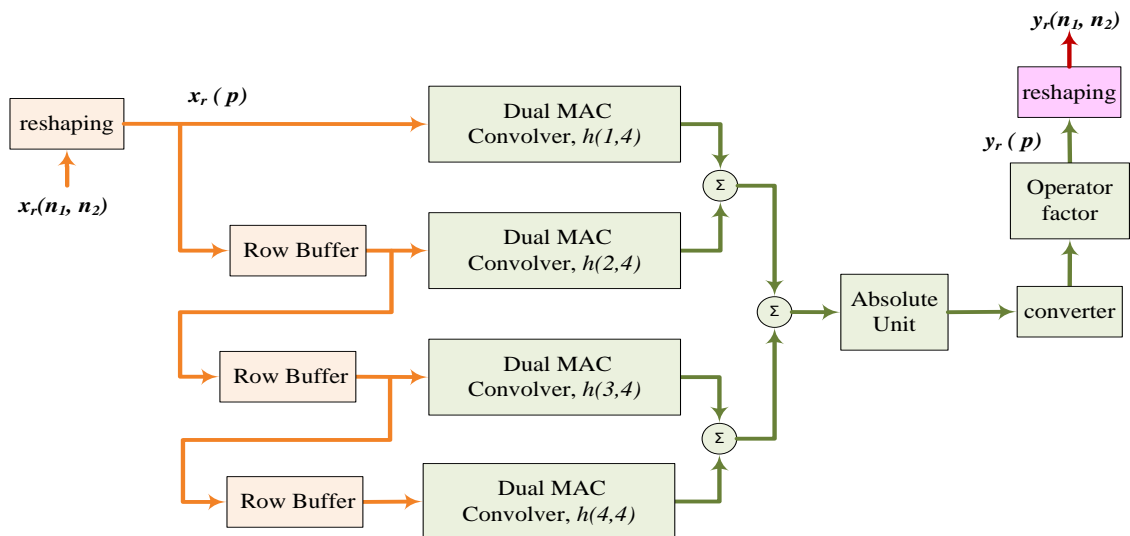
	2-D Single MAC convolver units			2-D Dual MAC convolver units		2-D Quad MAC convolver units	
	Unit 4 2x2	Unit 5 3x3	Unit 6 5x5	Unit 7 2x4	Unit 8 4x4	Unit 9 2x8	Unit 10 8x8
<b>FFs</b>	182	271	536	256	471	370	1 022
<b>LUTs</b>	125	167	355	178	337	273	807
<b>Slices</b>	46	92	144	73	123	111	272
<b>DSP 48E1s</b>	2	3	5	4	8	8	24
<b>RAMB 18E1s</b>	3	5	9	5	11	9	27

#### 4.4.2 Parallel 2-D Dual MAC Convolver Architectures

The parallel spatiotemporal convolution-filtering algorithm of Figure 4.8 can be realized by another FPGA-based architecture using the 2-D dual MAC convolver filter units of size  $(2 \times 4)$  and  $(4 \times 4)$ . Figure 4.13 and Figure 4.14 show  $(2 \times 4)$  and  $(4 \times 4)$  dual MAC operators for the spatiotemporal convolver architecture as unit 7 and unit 8 respectively. As shown in Table 4.2, unit 7 occupies almost half the logic area of unit 8.



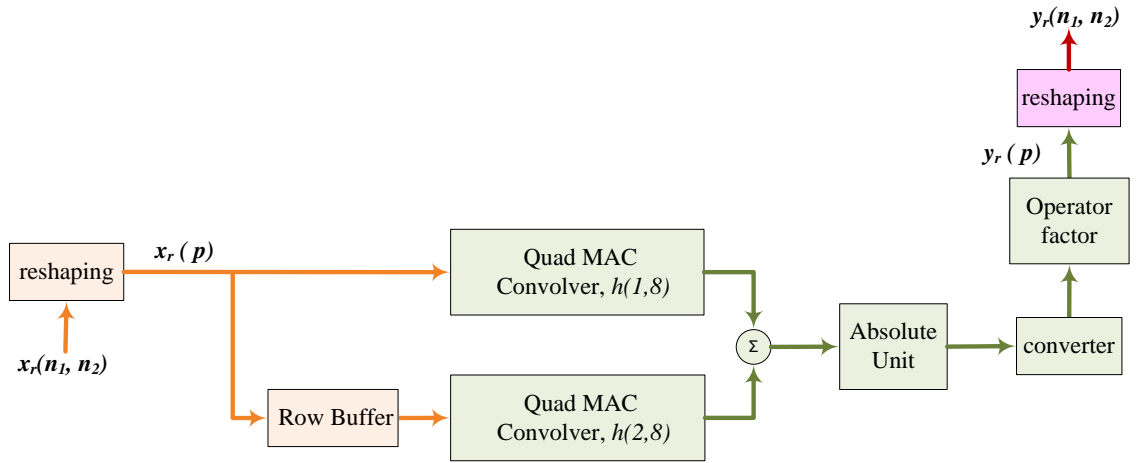
**Figure 4.13: Unit 7;  $(2 \times 4)$  Dual MAC convolver Architecture**



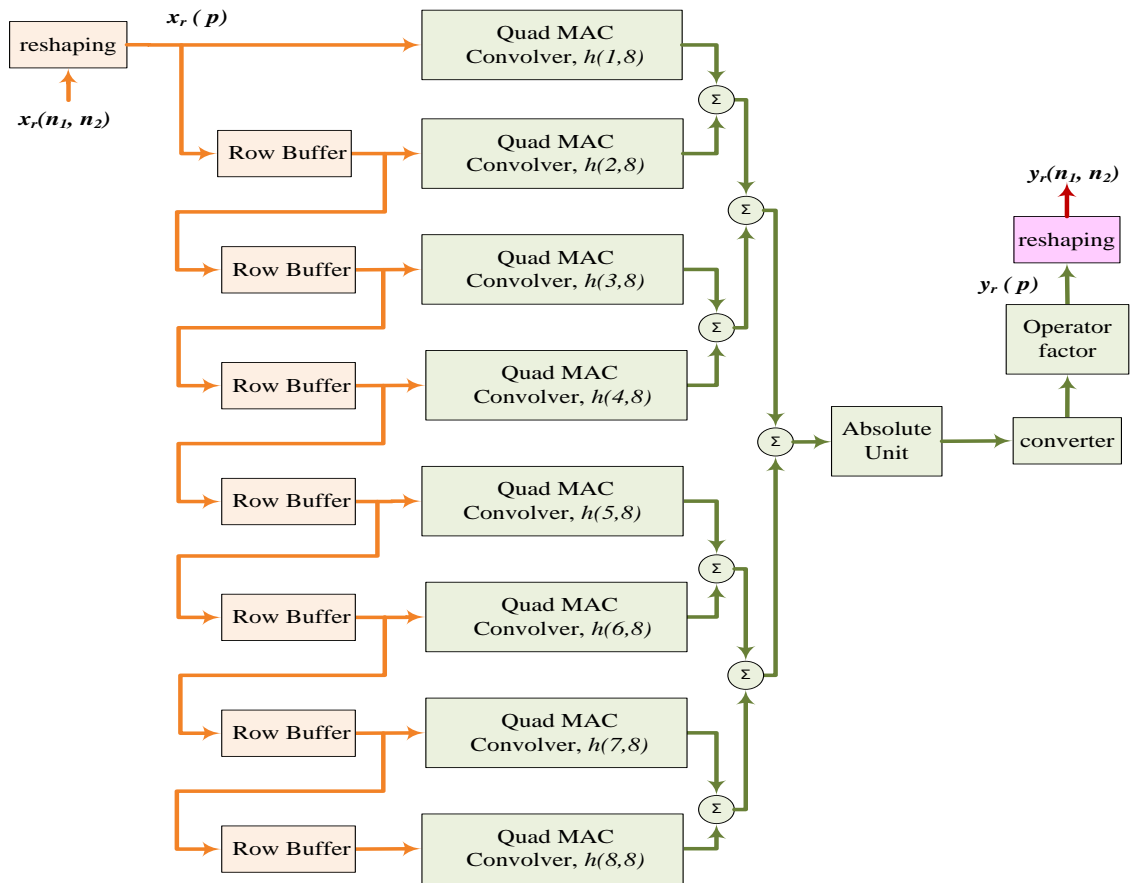
**Figure 4.14: Unit 8;  $(4 \times 4)$  Dual MAC convolver Architecture**

#### 4.4.3 Parallel 2-D Quad MAC Convolver Architectures

A third type of FPGA-based architectures for the parallel 2-D spatiotemporal convolution-filtering algorithm can be realized using 2-D quad MAC convolver filter units of size  $(2 \times 8)$  and  $(8 \times 8)$ , shown in Figure 4.15 and Figure 4.16, as unit 9 and unit 10 respectively. These units realize the 2-D MAC FIR operator of the filtering stage for the parallel linear image stream filtering algorithm of Figure 4.8. As shown in Table 4.2, unit 9 occupies almost one-third the logic area unit 10.



**Figure 4.15: Unit 9; (2×8) Quad MAC convolver Architecture**



**Figure 4.16: Unit 10; (8×8) Quad MAC convolver Architecture**

#### 4.4.4 Performance Indices of the Parallel Spatiotemporal Convolver Architectures

The performance indices of the seven 2-D spatiotemporal convolver architectures are considered as a complete set of area, speed, power, throughput and computation rate parameters using XSG to target the Virtex-6 ML605 board. The minimized utilized area of the seven architectures, as shown in Table 4.3, is due to the efficient implementation hierarchy of logic fabric, signals, I/O's and hard IPs such as Block RAMs or DSP blocks. These seven architectures are occupying proper logic area of FFs, LUTs, slices, DSP blocks and Block RAMs with the least area occupation for architecture 6 and the

largest for architecture 12; the seven architectures can be categorized by the area occupation starting from the smallest as architecture 6, 9, 11, 10, 7, 8 to the largest as architecture 12. Consequently, this area occupation affects the performance indices set of speed, power consumption, throughput and computation rate as shown in Table 4.4.

**Table 4.3: Logic Devices utilization by each of the parallel spatiotemporal filtering architectures**

	Archit. 6	Archit. 7	Archit. 8	Archit. 9	Archit. 10	Archit. 11	Archit. 12
2-D FIR filter kernel ( $M_1 \times M_2$ )							
	2x2	3x3	5x5	2x4	4x4	2x8	8x8
<b>FFs</b>	688	1 632	1 705	872	1 624	1 272	4 684
<b>LUTs</b>	442	1 132	1 108	588	1 170	947	3 722
<b>Slices</b>	205	441	502	283	520	426	1 421
<b>DSP 48E1s</b>	8	12	20	16	32	32	128
<b>RAMB 18E1s</b>	12	20	36	20	44	36	156

Several observations may be made from Table 4.4 performance indices. Firstly, the operating clock frequency is particularly insensitive to the occupied logic area [31] by the seven architectures, and principally operating around the indicated 214 MHz maximum frequency. Secondly, the dynamic power consumption at (40 nm) junction temperature of 54°C is monotonically decreases from 140mW for architecture 12, 46mW for architecture 10, 39mW for architecture 11, 27mW for architecture 8, 24mW for architecture 9, 21mW for architecture 7 down to 14mW for architecture 6. Thirdly, the highest throughput, in filtering a grayscale MRI slices, is achieved by architectures 6, 9 and 11, which is almost 2.25, 6, 2 and 4 times than that of architectures 7, 8, 10 and 12 respectively.

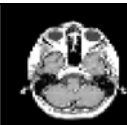
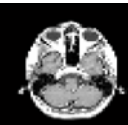












**Table 4.4: Performance indices of the parallel spatiotemporal filtering architectures**

	Archit. 6	Archit. 7	Archit. 8	Archit. 9	Archit. 10	Archit. 11	Archit. 12
2-D FIR filter kernel ( $M_1 \times M_2$ )							
	2x2	3x3	5x5	2x4	4x4	2x8	8x8
<b>Maximum Clock Frequency (MHz)</b>	215	214	216	215	213	214	213
<b>Dynamic Power (mWatt)</b>	14	21	27	24	46	39	140
<b>Throughput (FPS), 64x64 MRI</b>	54 931	24 414	8 789	54 931	27 465	54 931	13 732
<b>Throughput (FPS), 1024x1024 MRI</b>	214	95	34	214	107	214	53
<b>computation rate (GMACPS)</b>	1.8	2.7	4.5	3.6	7.2	7.2	28.8

According to (4.6), where the one frame time is inversely proportional to the size of the filtered image frame, thus, the throughput was significantly affected as the MRI size goes from  $64 \times 64$  to  $1024 \times 1024$  for the same 2-D kernel size. Although, the lowest throughput of (34 FPS) achieved by architecture 8 is still within the real-time performance of (30 FPS) frame rate [17, 112]. Consequently, the seven parallel spatiotemporal convolution architectures are evidently suitable for real-time applications [21, 34, 35, 38, 97-99, 102-104, 106, 108, 109, 111, 113, 114]. Fourthly, according to (4.8), the highest computation rate is accomplished by architecture 12, due to the three levels of parallelism. That is double than that of architecture 10 and 11. For the remaining implementations, that are (4), (3), (5) and (8) times than that of architecture 9, 8, 7 and 6 respectively.















The seven parallel spatiotemporal convolution architectures are developed as “plug and filter” architectures. Thus, twelve generic 2-D FIR filters can be used in the development of edge detection and noise smoothing for the biomedical imaging of  $64 \times 64$  greyscale MRI, then, improved as shown in Table 4.5 and Table 4.6 respectively.

**Table 4.5: Filtering results of  $64 \times 64$  greyscale MRI using architecture 8**

2-D FIR Filter	Generic 5x5 kernel	Filtered 64x64 MRI	2-D FIR Filter	Generic 5x5 kernel	Filtered 64x64 MRI
<b>Identity</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Identity</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
<b>EdgeXY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & -1 & 8 & -1 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Laplacian</b> $D.F = \left(\frac{1}{8}\right)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & -8 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
<b>EdgeX</b>	$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$		<b>PrewittX</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
<b>EdgeY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Blur</b> $D.F = \left(\frac{1}{16}\right)$	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	
<b>SobelXY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Smooth</b> $D.F = \left(\frac{1}{100}\right)$	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 5 & 44 & 5 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	
<b>SobelX</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -2 & 0 & 2 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Sharpen</b> $D.F = \left(\frac{1}{-16}\right)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 2 & -32 & 2 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
<b>SobelY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Gaussian</b> $D.F = \left(\frac{1}{52}\right)$	$\begin{bmatrix} 1 & 1 & 2 & 1 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 2 & 4 & 8 & 4 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 1 & 1 & 2 & 1 & 1 \end{bmatrix}$	

The MRI filtered images in Table 4.5 show the effect of the twelve generic 2-D filtering operators, eight edge detection operators and four noise smoothing operators. The eight edge detection operators are EdgeXY, EdgeX, EdgeY, SobelXY, SobelX SobelY, Laplacian and PrewittX. The four noise-smoothing operators used are Blur, Smooth, Sharpen and Gaussian. The Edge operators detect the changes or differences in the pixel value at the edges of regions, the contrast between the region and the background or between two regions. Thus, the Edge operators are differencing filters, which make them sensitive to noise, as in the first eight operators of Table 4.5, which act as low pass filters except the Laplacian filter. This high pass filter can be used to detect edges of all orientations, but is very sensitive to noise.

**Table 4.6: Filtering results for grayscale 64×64 MRI of twelve improved 2-D FIR filter operators using architecture 8**

2-D FIR Filter	Improved 5x5 kernel	Filtered 64×64 MRI	2-D FIR Filter	Improved 5x5 kernel	Filtered 64×64 MRI
Identity	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		Identity	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
EdgeXY D.F= $(\frac{1}{8})$	$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ -1 & -1 & 16 & -1 & -1 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$		Laplacian	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -3 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
EdgeX	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		PrewittX D.F= $(\frac{1}{3})$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
EdgeY	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		Blur D.F= $(\frac{1}{16})$	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 16 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	
SobelXY	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		Smooth D.F= $(\frac{1}{100})$	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 5 & 144 & 5 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	
SobelX D.F= $(\frac{1}{3})$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & -2 & 1 & 2 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		Sharpen D.F= $(\frac{1}{9})$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & -1 & 17 & -1 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
SobelY D.F= $(\frac{1}{3})$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		Gaussian D.F= $(\frac{1}{16})$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 2 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	

#### 4.5 Three Dimensions Application: Parallel Colour MRI Filtering

The greyscale MRI pixels are represented by a scalar value between zero and (255). The logical extension to the grayscale 2-D MRI is the Colour 3-D MRI or functional MRI (fMRI). The colour 3-D MRI pixels are represented by a vector of components rather

than a scalar. Then, colour 3-D MRI or fMRI is represented as a vector-base image of  $(N_1 \times N_2 \times 3)$  size, as following:

$$\mathbb{x}(n_1, n_2) = \begin{bmatrix} x_1(n_1, n_2) \\ x_2(n_1, n_2) \\ x_3(n_1, n_2) \end{bmatrix} \quad (4.14)$$

Where,  $x_1(n_1, n_2)$ ,  $x_2(n_1, n_2)$  and  $x_3(n_1, n_2)$  are the MRI three colour components of red, green and blue of size  $(N_1 \times N_2)$  each. The 3-D colour MRI or fMRI is typically represented by a three-dimensional vector of eight bits per component, resulting in a 24-bit colour system. The 3-D colour MRI filtering involves the 2-D convolution of the raw colour MRI  $\mathbb{x}(n_1, n_2)$  with a 2-D FIR operator  $h(m_1, m_2)$  to produce the three components filtered colour output  $\mathbb{y}(n_1, n_2)$ , given by,

$$\begin{bmatrix} y_1(n_1, n_2) \\ y_2(n_1, n_2) \\ y_3(n_1, n_2) \end{bmatrix} = \begin{bmatrix} x_1(n_1, n_2) \\ x_2(n_1, n_2) \\ x_3(n_1, n_2) \end{bmatrix} ** h(m_1, m_2) \quad (4.15)$$

where, \*\* is the 2-D convolution, which can be expressed as:

$$y(n_1, n_2) = \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} x(m_1, m_2) h(n_1 - m_1, n_2 - m_2) \quad (4.16)$$

For the real-time 3-D colour MRI filtering [104, 112], the parallel 2-D linear image filtering algorithm, as shown in Figure 4.7, and its FPGA implementation on Figure 4.8, is preferred for its faster filtering speed, high computation rate and higher throughput with low dynamic power consumption and small logic area occupation. Consequently, the parallel 2-D spatiotemporal convolver architectures 6, 7, 8, 9, 10, 11 and 12 can be utilized as FPGA implementation for the parallel 3-D colour MRI filtering applications. This parallel colour MRI convolution algorithm can be represented as follows:

$$y_r(n_1, n_2) = \sum_{r=0}^3 \left( \sum_{m_1=0}^{\frac{N_1-1}{2}} \sum_{m_2=0}^{\frac{N_2-1}{2}} \mathbb{x}_r(m_1, m_2) h(n_1 - m_1, n_2 - m_2) \right) \quad r = 0, 1, \dots, 3 \quad (4.17)$$

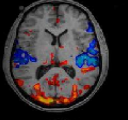
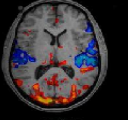


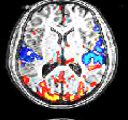
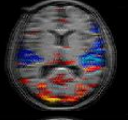
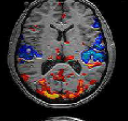




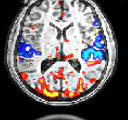
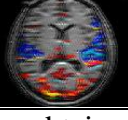
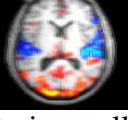
The performance indices of the seven architectures for the colour MRI filtering are the same as those indices for the greyscale MRI, as shown in Table 4.3 for the logic area occupation, and Table 4.4 for the maximum clock frequency, dynamic power consumption and the computation rate. However, the total throughput (FPS) is less due to the three frames of red, green and blue that constitutes the 3-D colour MRI. Thus, the total throughput for the parallel colour 3D MRI filtering is 1494, 664, 239, 1494, 747,



1494 and 373 using architectures 6, 7, 8, 9, 10, 11 and 12 respectively for an input colour MRI of (224×224×3) size.

The colour MRI filtered results using architecture 8 are shown in Table 4.7. Twelve 2-D FIR filters can be used in the development of edge detection and noise smoothing of the biomedical imaging of 224×224×3 colour MRI slice. These twelve MRI filtering operators are improved for the suitability of this particular biomedical imaging. The MRI filtering results are achieved by modifying the generic twelve 2-D FIR filters of Table 4.1 according to a heuristic criteria: If the sum of the ( $M_1 \times M_2$ ) elements is larger than the generic 2-D filter by 1 or more, then, the filtering result will be a brighter MRI. This criterion is applied to the MRI results of EdgeXY, EdgeX, EdgeY, SobelXY, SobelX, SobelY, Laplacian, PrewittX, Blur and Sharp operators.

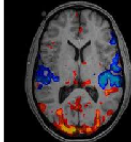
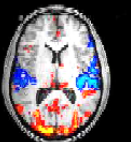
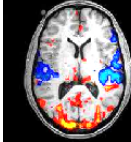
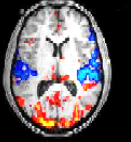
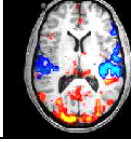
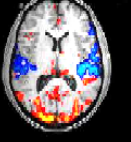
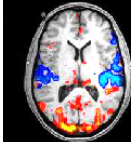
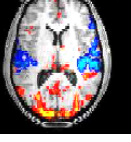
**Table 4.7: Filtering results for colour 224×224×3 MRI of twelve developed 2-D FIR filter operators using architecture 9**

2-D FIR Filter	Generic 5x5 kernel	Filtered Colour MRI	2-D FIR Filter	Generic 5x5 kernel	Filtered Colour MRI
<b>Identity</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Identity</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
<b>EdgeXY</b> $D.F = \left(\frac{1}{-8}\right)$	$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 16 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$		<b>Laplacian</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -3 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
<b>EdgeX</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>PrewittX</b> $D.F = \left(\frac{1}{3}\right)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
<b>EdgeY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Blur</b> $D.F = \left(\frac{1}{16}\right)$	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 16 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	
<b>SobelXY</b>	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 2 & 0 & -1 & 0 \\ 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Smooth</b> $D.F = \left(\frac{1}{50}\right)$	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 5 & 44 & 5 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	
<b>SobelX</b> $D.F = \left(\frac{1}{3}\right)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & -2 & 1 & 2 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Sharpen</b> $D.F = \left(\frac{1}{-8}\right)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 2 & -32 & 2 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
<b>SobelY</b> $D.F = \left(\frac{1}{3}\right)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		<b>Gaussian</b> $D.F = \left(\frac{1}{8}\right)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	

Conversely, a darker MRI will be obtained if the sum of ( $M_1 \times M_2$ ) elements is smaller than the generic 2-D filter, as with the MRI results of Smooth and Gaussian. If the ( $M_1 \times M_2$ ) elements sum is zero, the resulting MRI is not necessarily completely black, but it will be very dark, this is the case with the generic 2-D operators of EdgeXY,

EdgeX, EdgeY, SobelXY, SobelX, SobelY, Laplacian, PrewittX. If the 2-D operator elements sum is 1, then, the resulting MRI will have the same brightness as the original MRI slice, this last criterion is applicable to all the twelve 2-D operators. As a comparison of filtering results, the seven architectures are used, as shown in Table 4.8, to process the same colour MRI using the  $(M_1 \times M_2)$  2-D FIR operators of architectures 6, 7, 8, 9, 10, 11 and 12. The filtered MRIs are brightened due to the sum of the  $(M_1 \times M_2)$  elements being larger than the identity 2-D filter by 1.

**Table 4.8: Filtering results for colour  $224 \times 224 \times 3$  MRI using the parallel spatiotemporal filtering architectures**

2-D FIR Filter	Generic $(M_1 \times M_2)$ kernel	Filtered Colour MRI	2-D FIR Filter	Generic $(M_1 \times M_2)$ kernel	Filtered Colour MRI
Identity	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		2x4 Brightener	$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	
2x2 Brightener	$\begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$		4x4 Brightener	$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	
3x3 Brightener	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$		2x8 Brightener	$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	
5x5 Brightener	$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		8x8 Brightener	$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	

The filtering results within the above tables show the edge filtering, noise smoothing and edge enhancement for diverse regions of the colour MRI that can be used by the physician to noninvasively depict areas of the brain for investigating either the human brain prior to neurosurgery or the brain functional activities in detail that are used for specific tasks. Thus, architecture 6, 7, 8, 9, 10, 11 and 12 can be plugged and used to develop filters for many colour MRI or fMRI, current and future potential practical applications; these include, reading of brain states [113], intracranial Hemorrhage annotation [96], brain-computer interfaces [98, 114] and communicating with locked-in patients [115].

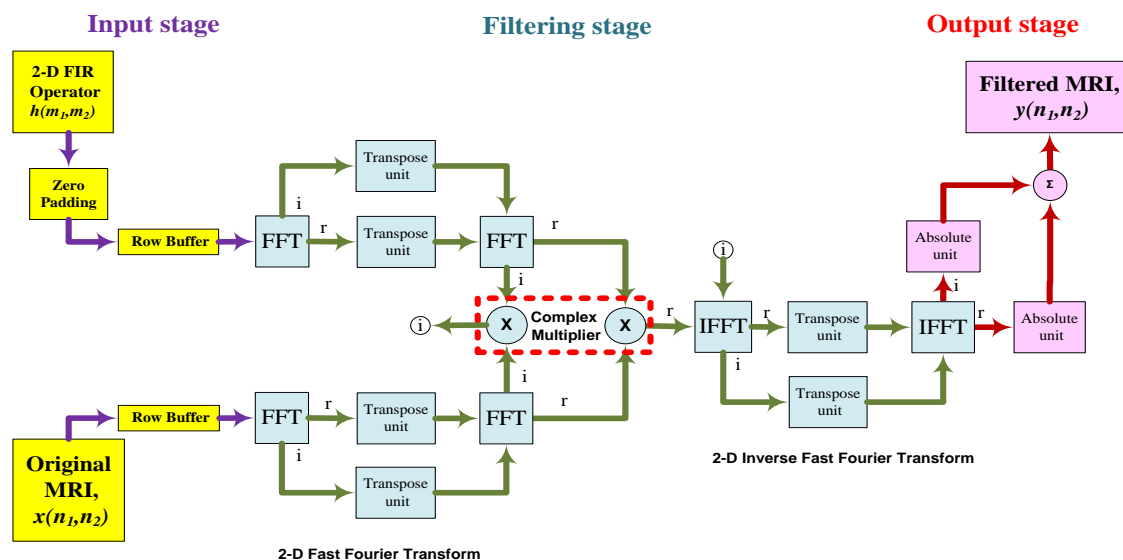
#### 4.6 FPGA Implementation as Indirect Parallel 2-D FFT Filtering Architectures

The parallel 2-D image filtering algorithms can process the 2-D image filtering applications [55] [32, 33] in which the 2-D FIR kernel length is longer than  $(10 \times 10)$

coefficients [82] [88, 92]. The FPGA implementation of this type of algorithm can be efficiently captured by more than one fast parallel 2-D FFT filtering architecture, depending on the 2-D image segmentation, FPGA memory and accordingly the degree of parallelism in the filtering stage. Consequently, two efficient hardware architectures are developed; single 2-D FFT convolution architecture and parallel 2-D FFT convolution architecture. The first approach is described in subsection 4.6.1 and 4.6.2 as architectures 13, and 14, respectively. The second approach is described in subsection 4.6.3 as architecture 15.

#### 4.6.1 Single 2-D FFT Filtering Unit Implementation

The 2-D FFT convolver engine shown in Figure 4.6 can be realized as a single 2-D FFT convolution architecture, shown in Figure 4.17. This architecture consists of input, filtering and output stages. Within the input stage the 2-D MRI and the 2-D FIR operator are streamed using the linear 2-D Stream Filtering Method, as explained in subsection 4.2.1. The temporal stream of the 2-D FIR operator is zero padded to be equal in size to the 2-D MRI. Both streams are row buffered to be processed in the filtering stage. The row buffer component is implemented as shown in Figure 4.12.

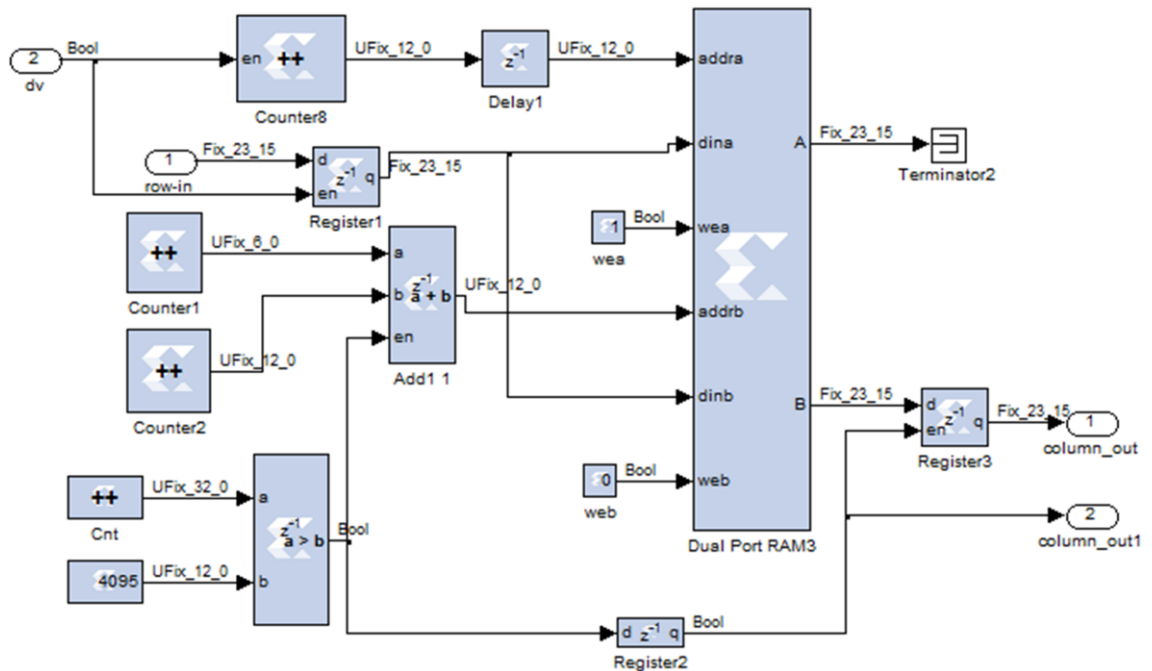


**Figure 4.17: Architecture 13; the implementation of the Fast Single 2-D FFT Filtering unit on the Virtex-6 FPGA board**

The 2-D filtering stage is a spatiotemporal to frequency transformation structure, as explained in subsection 4.2.4. This structure consists of 2-D FFT, complex multiplier and 2-D IFFT components. The 2-D FFT consists of two Xilinx FFT v7\_0 blocks and two transpose unit structure, as shown in Figure 4.17. The transpose unit can be realized as in Figure 4.18. Where, the transformed 2-D matrix is first inputted via port A of a

dual port RAM, and then accessed in port B by two address control circuitries. The writing address circuitry is an up counter from zero to  $(N_1 \times N_2 - 1)$  to store the image stream row by row. While, reading address circuitry is a combination of two up counter to generate the transpose 2-D matrix out of port B column by column.

The two 2-D matrices of the FIR coefficients and the MRI are simultaneously frequency transformed via the real inputs of two parallel Xilinx FFT v7\_0 blocks, and setting the imaginary input to zero. Each FFT outputs frequency spectrum in, real and imaginary parts. Thus, the image will be transformed into an intermediate image of real array and imaginary array. Subsequently, two transpose units are used. Next, 1-D FFT is repeated on each column of the intermediate image. The resulting real and imaginary parts are the image's 2-D frequency spectrum,  $X(k_1, k_2)$ .



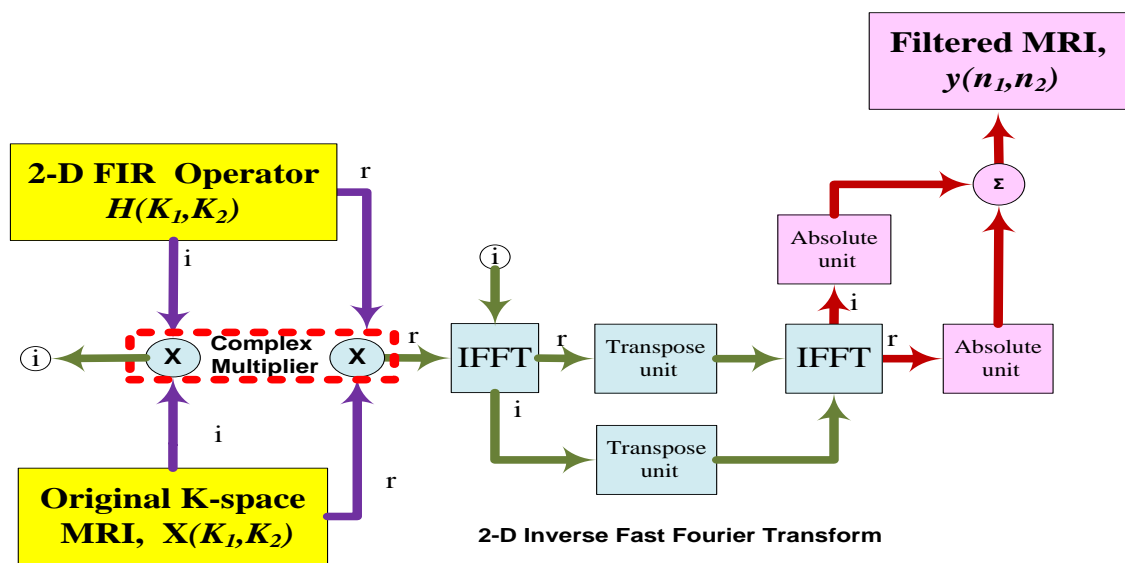
**Figure 4.18: transpose unit implementation**

Similarly, 1-D FFT pair calculate the 2-D FIR operator's 2-D frequency spectrum,  $H(k_1, k_2)$ . Using Xilinx complex multiplier block, these two frequency spectra are point-by-point multiplied to produce  $Y(k_1, k_2)$ , which represents the 2-D FFT of the filtered image. To transform back into the time domain, the inverse Fourier transform of  $Y(k_1, k_2)$  is calculated by taking the 1-D inverse FFT of each column, followed by two transpose units to arrange the two filtered image components, then the 1-D inverse FFT of each row is processed.

In the output stage, the two streams of the filtered MRI are conditioned by an absolute unit first, and then summed to produce the 2-D filtered MRI  $y(n_1, n_2)$ . The final reconstructed output is connected to a gateway-out block, which provides the conversion from the fixed point format which is used by the FPGA to floating point numerical representation used by Simulink blocks for displaying the filtered MRI.

#### 4.6.2 Single 2-D IFFT Convolution k-Space MRI Unit Implementation

MRI slice visualization is achieved by reconstructing the image from k-space MRI data [116-118]. The MRI machine collects data in the frequency domain which is known as the k-space. This real-time data needs to be filtered first to create the digital 2-D MRI that can be accurately inspected. This filtering can efficiently be processed using 2-D IFFT convolution algorithm. Due to the k-space MRI data being collected as 2-D frequency data, then the 2-D IFFT part of architecture 14, as shown in Figure 4.19, can be used for filtering and visualization. Where, the two frequency inputs  $X(K_1, K_2)$  and  $H(K_1, K_2)$  are point-by-point complex multiplied.



**Figure 4.19: Architecture 14; implementation of the Fast Single 2-D IFFT convolution k-space MRI Filtering unit on the Virtex-6 FPGA board**

Then, the resultant real and imaginary 2-D filtered MRI has to be transferred back to the spatiotemporal domain by 2-D IFFT and conditioned to be visualized as a digital 2-D MRI image. The FPGA implementation details are as explained in the previous subsection of 4.6.2.

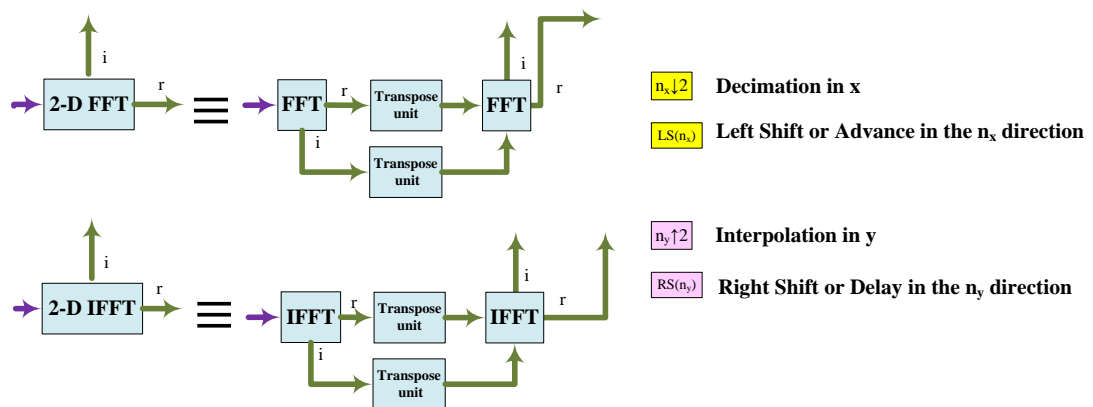
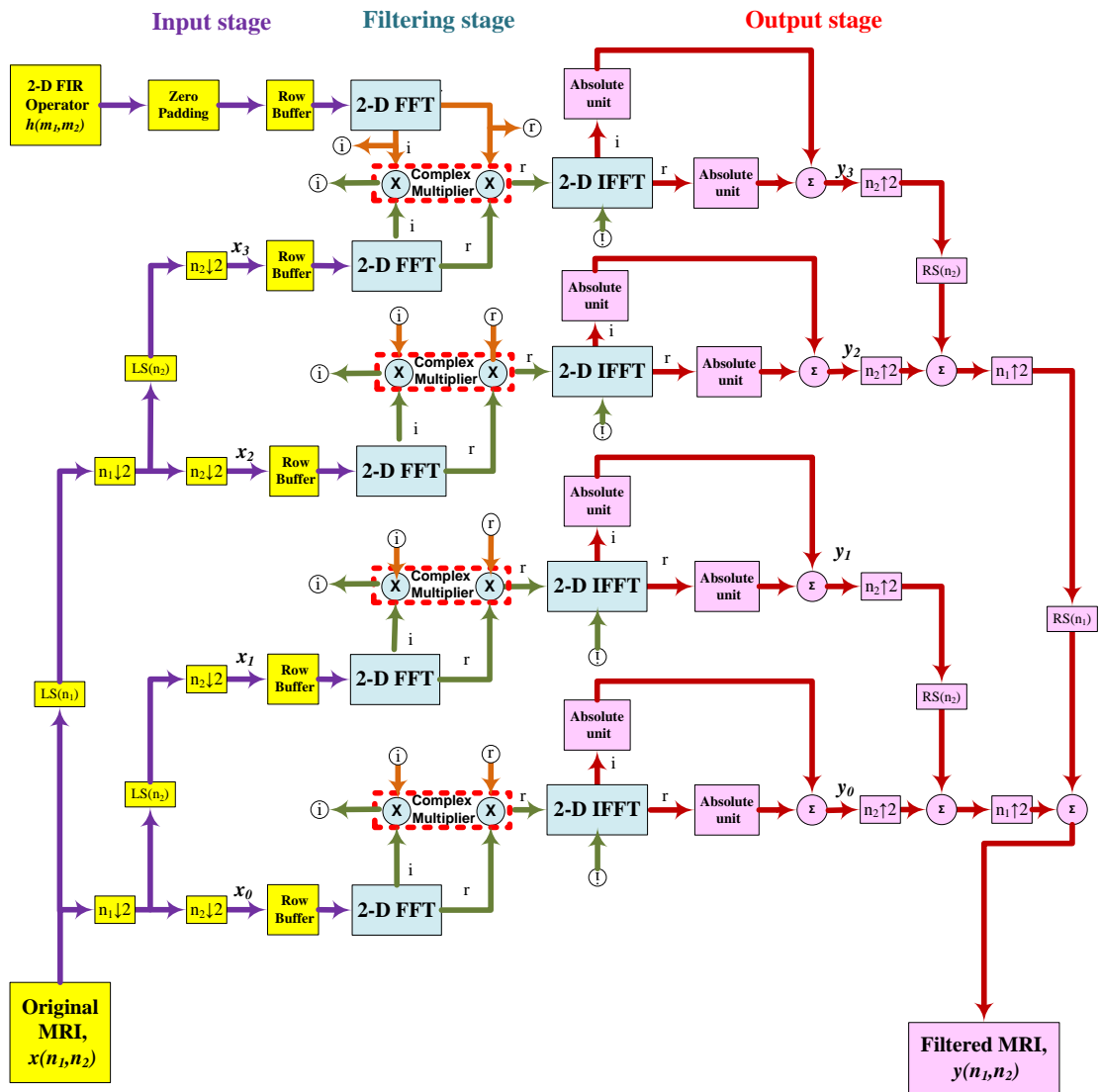
### 4.6.3 Fast Parallel 2-D FFT Filtering Architecture

The memory limitation of any digital signal processor to store the input images of larger sizes necessitates the segmentation of the 2-D input image into independent 2-D sub-images. Consequently, a parallel fast convolution algorithm has to be developed to simultaneously process these sub-images. Thus, the generalized parallel 2-D filtering algorithm of Figure 4.7 may be realized in hardware as architecture 15, shown in Figure 4.20. This architecture consists of input, filtering and output stages. The input decimated by 2 stage and the output interpolated by 2 stage are as described in subsection 4.3.1 and 4.3.3 respectively. The main difference is in the filtering stage which in this case based on the convolution property of the 2-D FFT.

The input MRI is decimated by 2 to produce four independent sub-MRI blocks  $x_0$ ,  $x_1$ ,  $x_2$  and  $x_3$ . These sub-images and the 2-D FIR operator are converted from spatial parallelism to temporal streams as described in subsection 4.2.1. Each stream is row buffered before being processed within the filtering stage. Due to the independency of the four decimated sub-streams, the fast 2-D filtering stage is carried out simultaneously using parallel 2-D FFT convolution array. Consequently, there are no internal communications in the convolution-filtering array due to the elimination of boundary conditions.

The filtering stage consists of four parallel 2-D Fast Fourier Transforms (FFT) convolution structure. Each 2-D FFT convolution structure is fast single 2-D FFT convolution sub-architecture, as described in subsection 4.6.1, where, 2-D spatial convolution is achieved by parallel complex multiplication. The four sub-MRI pixels streams and the 2-D FIR operator are frequency transformed by 2-D FFT, complex multiplied and spatially transformed by 2-D inverse FFT. The resultant filtered sub-MRIs have two components; real and imaginary. Each component is conditioned by an absolute unit and summed to produce four decimated by 2 filtered sub-images of  $y_0$ ,  $y_1$ ,  $y_2$  and  $y_3$ , as shown in Figure 4.20.

The final filtered 2-D MRI is reconstructed from the four sub-MRI using interpolation by 2. This output signal is connected to a gateway-out block, which provides the conversion from the FPGA-used fixed point format to floating point numerical representation used by Simulink blocks for displaying the filtered 2-D MRI.



**Figure 4.20: Architecture 15; implementation of Fast Parallel 2-D FFT convolution Algorithm on the Virtex-6 FPGA board**

#### 4.6.4 Performance Indices of Parallel 2-D FFT Convolution Architectures

The fast 2-D FFT convolution architectures results are presented as logic devices utilization tables, performance indices tables and 2-D MRI filtered images tables. The performance indices of three 2-D FFT convolver architectures are considered as a

complete set of area, speed, power and throughput using XSG to target the Virtex-6 ML605 board.

The three main variables that affect the 2-D image filtering results are the image size, 2-D FIR operator size and accordingly the size of the N-point FFT/IFFT. Hence, comparative evaluation results of the three FPGA implementation architectures are obtained for greyscale MRI slices of  $32 \times 32$ ,  $64 \times 64$ ,  $128 \times 128$ ,  $256 \times 256$  and  $512 \times 512$  size and distinctive 2-D FIR operator of  $15 \times 15$ ,  $31 \times 31$ ,  $63 \times 63$ ,  $127 \times 127$  and  $256 \times 256$  coefficients, to be individually applied with a (32, 64, 128 and 256) N-point FFT/IFFT. Accordingly, the area occupation, performance indices and selected filtered outputs of the two architectures 13 and 15 are indicated in Table 4.9, Table 4.10, Table 4.11, Table 4.12, Table 4.13 and Table 4.14 respectively. Moreover, the performance indices of architecture 14 are compared to that of architecture 13, and the filtering results are presented in Table 4.15 for two k-space MRI slices.

**Table 4.9: Logic Devices utilization by architecture 13**

Greyscale MRI size ( $N_1 \times N_2$ )	2-D FIR kernel ( $M_1 \times M_2$ )	FFT N-point	FFs	LUTs	Slices	DSP 48E1s	RAMB 18E1s
32×32	15×15	32	8826	4837	1844	96	16
64×64	31×31	64	10609	5830	2065	108	31
128×128	63×63	128	13343	7533	2900	148	84
256×256	127×127	256	15315	9087	3568	160	148

**Table 4.10: Logic Devices utilization by architecture 15**

Grayscale MRI size ( $N_1 \times N_2$ )	2-D FIR kernel ( $M_1 \times M_2$ )	FFT N-point	FFs	LUTs	Slices	DSP 48E1s	RAMB 18E1s
64×64	15×15	32	35174	19366	7555	384	64
128×128	31×31	64	42299	22970	9203	432	124
256×256	63×63	128	53208	30123	11643	592	336

The minimized logic area of the two architectures, as indicated in Table 4.9 and Table 4.10, are occupying proper logic area of FFs, LUTs, slices, DSP blocks and Block RAMs. Where, the single architecture 13 was on average occupying less than one-third logic fabric, one-quarter of DSP blocks and half BRAMs than that of the parallel architecture 15. Consequently, this area occupation affects the performance indices set of speed, power consumption and throughput as indicated in Table 4.11 and Table 4.12. These tables present three performance indices of frequency, power and throughput.

Several observations can be made from these results. Firstly, the maximum clock frequency is steadily speeding up [33] as the MRI size, 2-D operator size and FFT N-



point are decreasing. Secondly, the power consumption monotonically decreases as the MRI size, 2-D operator size and FFT N-point are decreasing. Thirdly, architecture 13 outperformed architecture 15 in maximum clock frequency and power consumption. Fourthly, architecture 15's throughput outperforms that of architecture 13, on average, by four times within all the corresponding MRI size, 2-D operator size and FFT N-point.

Architecture 14 utilized 2-D IFFT part of architecture 13 to filter real-time k-space MRI data. Architecture 14 occupied less logic area than architecture 13 by (31%) FFs, (5%) LUTs, (17%) slices, (75%) dedicated DSP 48E1s multiplier, and (14%) RAMB 18E1s block memory. Consequently, architecture 14 consumes less dynamic power of 108 mW at maximum clock frequency of 190 MHz .

**Table 4.11: Performance indices of architecture 13**

Grayscale MRI size ( $N_1 \times N_2$ )	2-D FIR kernel ( $M_1 \times M_2$ )	FFT N-point length	Maximum clock frequency (MHz)	Dynamic Power (mWatt)	Throughput (FPS)
32×32	15×15	32	260	59	25391
64×64	31×31	64	244	82	4964
128×128	63×63	128	229	121	998
256×256	127×127	256	203	282	193

**Table 4.12: Performance indices of architecture 15**

Grayscale MRI size ( $N_1 \times N_2$ )	2-D FIR kernel ( $M_1 \times M_2$ )	FFT N-point length	Maximum clock frequency (MHz)	Dynamic Power (mWatt)	Throughput (FPS)
64×64	15×15	32	229	255	22363
128×128	31×31	64	211	354	4292
256×256	63×63	128	190	499	828

In 2-D MRI filtering, an ( $N_1 \times N_2$ ) image can be enhanced by convolving the original image with a 2-D sharpening operator of ( $M_1 \times M_2$ ) kernel. A generic sharpening operator consists of ( $M_1 \times M_2$ ) coefficients of (-1), except the central element (m) to be calculated by the following equation:







$$m = (-2) \times (-1) \times M_1 \times M_2 - 1 \quad (4.18)$$

All elements are divided by an operator factor ( $s = M_1 \times M_2$ ) to ensure that the low spatial frequency is not amplified. The 2-D MRI filtering results are shown in Table 4.13 and Table 4.14 using architecture 13 and 15 respectively. The size of the original greyscale MRI is of 64×64, 128×128, 256×256 and 512×512 to be edge sharpened using a sharpen operator of 15×15, 31×31, 63×63 and 127×127 kernels respectively. The sharpen operator is a noise smoothing operators and Edge filtering without the







strong sensitivity to noise to prevent the output value being different from the input in uniform regions of the MRI.

Architecture 15 can filter MRI slices of double size than that filtered using architecture 13, due to the input MRI's decimation by 2. Then, the 2-D operator kernel and the N-size FFT has half the value of that used in architecture 13 for the same MRI size. The filtered MRIs are slightly larger by  $M-1$  pixels with the new two dimensions of  $(N_1+M_1-1) \times (N_2+M_2-1)$ .

**Table 4.13: the filtering results for greyscale  $(N_1 \times N_2)$  MRI of the generic 2-D FIR Sharpen operators using architecture 13**

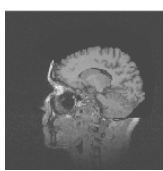

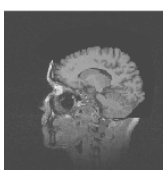

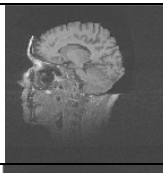

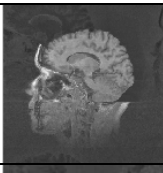

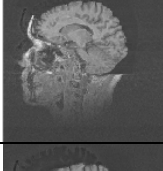

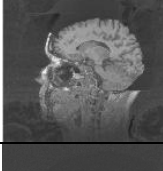

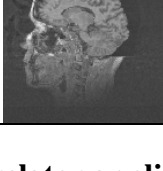

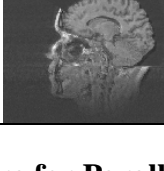
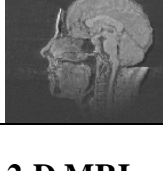
Grayscale MRI size $(N_1 \times N_2)$	2-D Sharpen kernel $(M_1 \times M_2)$	FFT N-point Size	Original $(N_1 \times N_2)$ MRI	Filtered MRI
64×64	31×31	64		
128×128	63×63	128		
256×256	127×127	256		

**Table 4.14: the filtering results for greyscale  $(N_1 \times N_2)$  MRI of the generic 2-D FIR Sharpen operators using architecture 15**

Grayscale MRI size $(N_1 \times N_2)$	2-D Sharpen kernel $(M_1 \times M_2)$	FFT N-point Size	Original $(N_1 \times N_2)$ MRI	Filtered $(N_1 \times N_2)$ MRI
128×128	31×31	64		
256×256	63×63	128		
512×512	127×127	256		

Architecture 14 was used to filter and visualize 2-D k-space MRI data input. This k-space data are generally obtained after processing data produced by an MRI machine. Architecture 14, when processing the real-time collected MRI data, was producing a sagittal view of a human head as shown in Table 4.15, where the original slice number 48 and slice 58, as shown on the top of the filtered results, are filtered and visualized out of a  $(256 \times 256 \times 99)$  k-space MRI volume data stack of slices. Because of the way the data were collected there is some spatial aliasing in the reconstructed image for the Sharpen operator's kernel below  $127 \times 127$  coefficients.

**Table 4.15: Sharpening results for k-space greyscale  $(256 \times 256)$  MRI of the generic 2-D FIR Sharpen operators using architecture 14**

Sharpen kernel $(M_1 \times M_2)$			Sharpen kernel $(M_1 \times M_2)$		
3x3			31x31		
7x7			63x63		
15x15			127x127		

#### 4.7 Cross-correlator application: FPGA Architecture for Parallel 2-D MRI Matched Filtering Algorithm

The detection of a targeted MRI slice from a MRI stack library is presented for its diagnosis applicability importance to facilitate the similarity access of a particular case from a pre-stored bank of images [119-121]. This similarity measure can be achieved by cross-correlation function [111], or referred to as 2-D image match filter. To perform cross-correlation by using convolution, the target image needs to be reversed to counteract the reversal that occurs during convolution. Thus, the 2-D MRI match filtering can be implemented using either architecture 13 or 15. To convert the target MRI slice into a 2-D match filtering operator, the targeted MRI slice must be rotated  $(180^\circ)$ , which is the same as being flipped left-for right and then flipped top-for-bottom.

Therefore, architecture 13 and architecture 15 can effectively be modified to be the implementation of the single correlator engine unit and the parallel 2-D MRI matched filtering algorithm respectively, to detect a targeted MRI slice  $h(m_1, m_2)$  within a MRI stack library  $x(n_1, n_2)$ ; where, the FIR impulse response coefficients of the filter are replaced by the samples of the target MRI slice input to be detected. Except, the target MRI slice has been flipped left-for right and then flipped top-for-bottom to implement its reversed-time version  $h(-m_1, -m_2)$ . For example, when  $x(n_1, n_2)$  and  $h(m_1, m_2)$ , are convolved to produce  $y(u)$  using architecture 15, the equation will be as:

$$y(u) = \sum_{r=0}^{k-1} \left( \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} h(n_1 - m_1, n_2 - m_2) x(m_1, m_2) \right) \quad (4.19)$$

Where,  $r=0, 1, \dots, 3$  are the parallel processing stages. In comparison, the correlation  $z(u)$  of  $x(n_1, n_2)$  and  $h(m_1, m_2)$  can be obtained by convolution of  $x(n_1, n_2)$  and  $h(-m_1, -m_2)$  as follows:

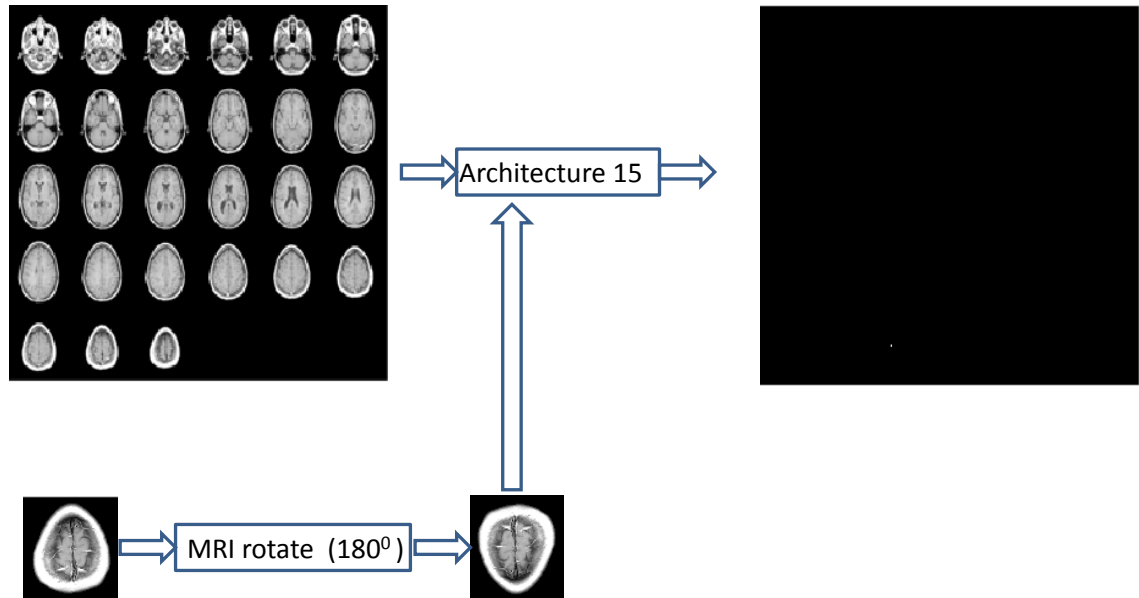
$$z(u) = \sum_{r=0}^{k-1} \left( \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} h(n_1 - (-m_1), n_2 - (-m_2)) x(m_1, m_2) \right) \quad (4.20)$$

That is, flipping left-for right in the  $N_1$  dimension and then flipped top-for-bottom in the  $N_2$  dimension is accomplished by reversing the sign of the time index. Accordingly, the parallel 2-D convolution described in (4.10) and (4.11) may be modified to perform parallel 2-D correlation filtering using (4.20).

Architecture 15 can be exploited to implement the parallel 2-D cross-correlation algorithm as the parallel 2-D MRI matched filter architecture with higher throughput. In the input stage, the original (512×512) MRI library is decimated by 2 to be divided into four (256×256) sub-MRI libraries, while the targeted MRI is of size (85×79). Hence, the N-point FFT/IFFT size is parameterized to be equal to 256, and, the targeted MRI slice must be zero padded to the size of (256×256) pixels.

The implementation of the parallel 2-D MRI match filtering algorithm is realized as architecture 15. The value of each pixel in the final correlated image is a measure of how well the target MRI slice matches the searched MRI library at that point. In this particular example, simply locating the brightest pixel in the final correlated MRI would specify the detected coordinates of the targeted MRI, as shown in Figure 4.21. The peak in the pixels values is separated from the 2-D cross-correlation output by thresholding.

For this example the thresholding level was 248, and all the pixels value below that level is set to zero, hence the black colour background.



**Figure 4.21: The parallel 2-D MRI matched filtering (cross-correlation) example using architecture 15**

#### 4.8 Conclusion

A parallel 2-D filtering algorithm was presented and mathematically modelled, to cover the extensive range of the linear 2-D image filtering and the 2-D image modality, in particular large MRI slice size. Then, a generic parallel hardware version of this parallel filtering algorithm was implemented on Virtex-6 FPGA board for the spatiotemporal MAC FIR and FFT convolution. The wide-ranged linear 2-D FIR kernels was split into two ranges. Firstly, those shorter than about  $10 \times 10$ , the spatiotemporal convolution was developed. Secondly, the FFT convolution was developed for the longer filter kernels. Ten generic architectures were devised. The performance indices for the ten architectures were considered as a complete set of area, speed, power and throughput. The superiority of the developed architectures were indicated by the minimized utilized area, high throughput, stable maximum clock frequency and low dynamic power consumption. Three successful applications on medical image filtering and detection were presented to demonstrate the high performance of the parallel 2-D convolution architectures.

# Chapter 5. Parallel 3-D Greyscale/Colour Image Filtering Algorithm and Its FPGA Implementations

## 5.1 Introduction

Three dimensional volumetric greyscale/colour image acquisition by means of various imaging modalities introduces degradation that affects the image quality [19, 114]. Such multi-dimensional image degradations include digitization noise, artifacts introduced during collection, video sequence motion blur, modality-inherent low contrast and three dimensional image transmission through noisy communication channels. Improving the three dimensional image quality is of utmost importance to help the end-user understand image better and allow the subsequent multi-dimensional image processing and analysis operations to benefit from such enhancement.

Thus, multi-dimensional image quality enhancement through noise suppression, blur/degradation removal and edge/contrast enhancement necessitate the proposal of multi-dimensional image filtering algorithm. This, however, requires a huge amount of volumetric image data and intensive computations to be performed to produce multi-dimensional filtering under strict time conditions. Consequently, this demand for high processing power in multi-dimensional applications cannot be achieved using sequential processing system only, but by a parallelized processing system. Practically, there are a large number of applications embracing parallel filtering to accomplish satisfactory throughput in reasonable time, for example robot vision [122, 123], medical image processing [124], weather forecasting [71], seismic data processing [125], video coding and processing [44, 67, 79] and wireless communication system [52].

Therefore, the aim of this chapter is to develop a performance-efficient parallel three dimensional greyscale/colour image filtering algorithm that exploits the multidimensional FIR filter kernels, increases the throughput by volumetric image decimation, and then reduces the processing time and the overhead of overlapping segments/ boundary conditions in the block filtering algorithm. The proposed parallel algorithm relieves the huge volume input size restriction covering the full range of the FIR filter kernels. The mathematical model for the parallel volume algorithm is presented and eight new generic architectures are implemented on Virtex-6 development board. New applications for four dimensions MRI (fMRI) volume data, real-time k-space MRI volume data and 3-D MRI cross-correlation are successfully

developed and the performance indices results as a complete set of area, speed, dynamic power and throughput are evaluated

The outline of this chapter is organised as follows: In section 5.2, the main research concepts are defined and mathematically explained. In section 5.3, a generalized parallel 3-D linear image filtering algorithm is presented and mathematically modelled. In section 5.4, five parallel 3-D spatial convolution architectures are developed, and their performance indices are evaluated as a complete set of area, speed, dynamic power consumption and throughput as well as the computation rate. In section 5.5, a successful application of four dimensions colour MRI slice filtering using the five parallel MAC architectures is developed, analysed and discussed. In section 5.6, three generic FFT architectures are developed, then, their performance indices are evaluated as a complete set of area, speed, dynamic power consumption and throughput. In section 5.7, another successful application of 3-D cross-correlation is realized for parallel 3-D MRI match filtering algorithm. Finally, the conclusion of this chapter is presented in section 5.8.

## **5.2 Research Concepts Definitions**

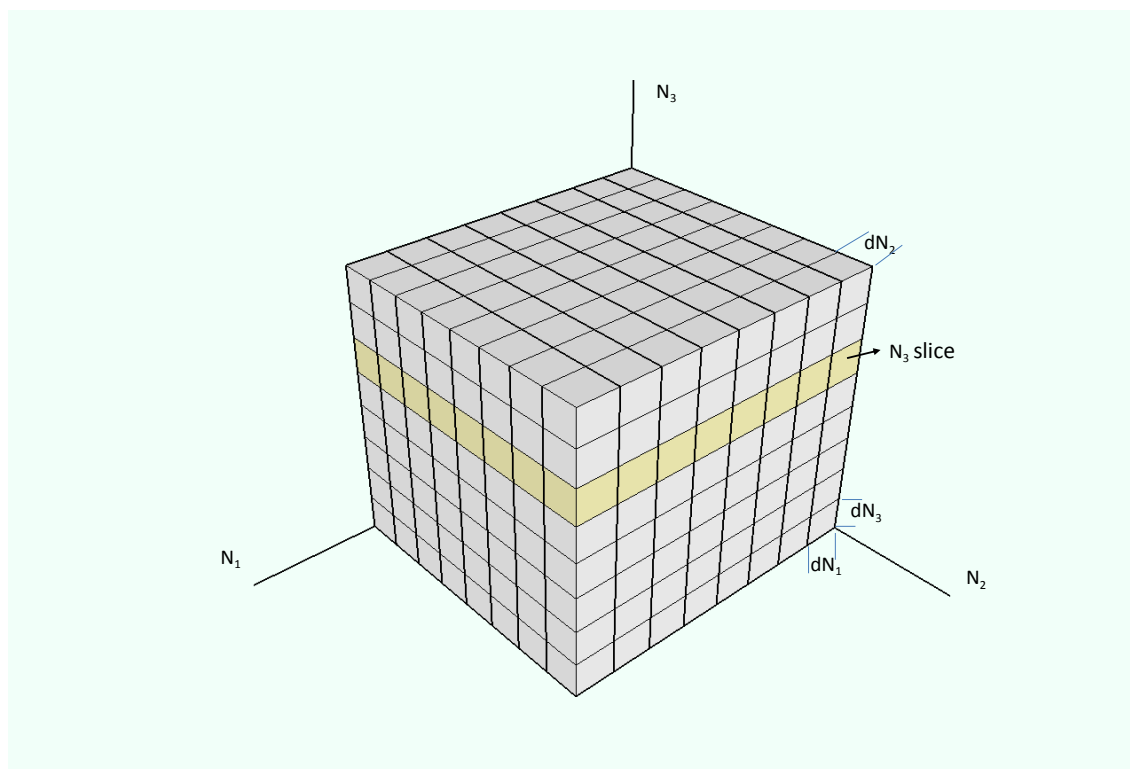
This section presents the main research concepts on which this chapter is based.

### **5.2.1 Linear 3-D Stream Filtering Method**

A 3-D image volume can be considered as 3-D matrix  $x(n_1, n_2, n_3)$  of size  $(N_1 \times N_2 \times N_3)$ . Where,  $n_1$ ,  $n_2$  and  $n_3$  are the row, column and slice (image frame) coordinates respectively, as shown in Figure 5.1. Each voxel has a 3-D size of  $(dN_1 \times dN_2 \times dN_3)$  pixels. For example, the 3-D MRI volume is formed as a series of 2-D MRI slices. The 2-D MRI slices are obtained along the  $N_1$ - $N_2$  object plane, and then the MRI detector is moved along the  $N_3$ -axis to acquire a new MRI slice. Usually voxel values are represented with 8-bit accuracy. This means their values range from zero (black) to 255(white), with intermediate values representing shades of gray. A typical 3-D MRI volume of 20 slices of  $256 \times 256$  pixels sums up to a total of 10.4 megabits. Thus, a 3-D image involves a large amount of data, which requires a huge storage capacity and reasonable speed to handle.

Linear stream 3-D image filtering is a logical extension of the linear stream 2-D image filtering. Nonetheless, the third dimension introduces a significant increase in the algorithm computation requirements which necessitates the need for a computationally efficient solution. 3-D Stream filtering can overcome this bottleneck in four steps to convert from spatial to temporal parallelism. Firstly, decimate the 3-D image volume

into 3-D volumetric sub-image. Secondly, decompose the decimated 3-D sub-images into frames of 2-D slices. Thirdly, partition each 2-D slice into blocks of rows. Fourthly, convert spatially partitioned blocks of 2-D slices into temporal parallelism.



**Figure 5.1:3-D image physical representation.**

An image is often streamed at a rate of one pixel per clock cycle. The parallel 3-D filtering operations are performed on the fly in parallel for the pixels using a separate 3-D FIR filter on each partition block. Architectures 16, 17, 18, 19, 20, 21, 22 and 23 are implemented to achieve these four levels of parallelism. Thus, a linear 3-D stream filtering method exploits as much processing as required on the streamed pixels as passing through the FPGA.

### **5.2.2 Linear 3-D FIR Filters**

The 3-D FIR filters [43, 126, 127] are neighbourhood-based operators on three dimensional kernels. That is the intensity of each voxel is enhanced according to the intensities of the neighbouring voxels based on the 3-D kernel size. A 3-D FIR filter is an operator of three 2-D consecutive MAC FIR operators. These operators map a 3-D image into a filtered 3-D image by a 3-D accumulated sum of point by point multiplication of a 3-D FIR impulse response coefficients,  $h(m_1, m_2, m_3)$ , and the voxel values of the 3-D image,  $x(n_1, n_2, n_3)$ , within the 3-D window [19]. This 3-D FIR filter interacts with its 3-D image input through a linear 3-D convolution process:



$$y(n_1, n_2, n_3) = x(n_1, n_2, n_3) *** h(m_1, m_2, m_3) \quad (5.1)$$

Where, \*\*\* is the linear 3-D convolution operation,  $h(m_1, m_2, m_3)$  is the 3-D operator of size  $(M_1 \times M_2 \times M_3)$  and  $x(n_1, n_2, n_3)$  of size  $(N_1 \times N_2 \times N_3)$ . Linear 3-D filtering is equivalent to performing a 3-D convolution in the spatial domain with the kernel flipped left-for-right, top-for-bottom and frame-for-frame,  $h(-m_1, -m_2, -m_3)$ , thus the above equation can formally stated as:

$$y(n_1, n_2, n_3) = \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} \sum_{m_3=0}^{M_3-1} x(m_1, m_2, m_3) h(n_1 - m_1, n_2 - m_2, n_3 - m_3) \quad (5.2)$$

Since the linear 3-D FIR filters can be designed to have symmetrical coefficients around the vertical, horizontal and the frame axes, then these flips are already achieved. Thus, the 3-D convolution can be achieved by 3-D FIR filter of coefficients' symmetry in the three dimensions.

The values and size of the 3-D FIR impulse coefficients determine the filtering operations and the convolution implementation domain [19, 44, 82, 128]. Some of the common filtering operations are edge detection and noise smoothing depending on the following weights sum equation:

$$\sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} \sum_{m_3=0}^{M_3-1} h(m_1, m_2, m_3) = \begin{cases} 0 \\ 1 \end{cases} \quad (5.3)$$

The coefficients sum is zero or one for the 3-D edge detection filters or 3-D noise smoothing filters respectively. Some of the common generic 3-D edge filters are Edge, Sobel, Laplacian and Prewitt, and for the noise smoothing filters, Sharpen, Gaussian, Smooth and Blur, as shown in Table 5.1. These 3-D filters will be implemented and applied to the biomedical imaging of Magnetic Resonant Imaging (MRI) as a “plug and develop” filtering processor. These 3-D image operators are realized in the spatial domain as architectures 16, 17, 18, 19 and 20, or in the frequency domain as architectures 21, 22 and 23.

A 3-D filter operator size  $(M_1 \times M_2 \times 3)$  determines the proper convolution implementation for the linear 3-D filter to be in the spatial or frequency domain. A spatial convolution is more important of these two, since volumetric images have their information encoded in the spatial domain rather than the frequency domain [82]. Where, the 3-D volume image input is  $x$  having  $n_x$  number of pixels, and a 3-D FIR

filter kernel is  $h$  having  $n_h$  number of elements. Then the straight spatial convolution method required  $O(n_x n_h)$  operations, while a FFT convolution perform the 3-D convolution in  $O(n_x \log_2 n_x)$  steps [44]. Since,  $n_h$  is always greater than  $\log_2 n_x$  for long 3-D kernel, then, the Fourier multiplication technique should quickly win over spatial multiplication for long FIR kernels. For short 3-D kernel, however, the inequality formula will be  $n_h < \log_2 n_x$ , hence, the implementation should be carried out using the 3-D spatial convolution. Consequently, for filter kernels shorter than about  $10 \times 10 \times 3$ , the direct convolution can be implemented, but not limited to, as  $2 \times 2 \times 3$ ,  $3 \times 3 \times 3$ ,  $2 \times 4 \times 3$ ,  $4 \times 4 \times 3$  and  $2 \times 8 \times 3$  kernel architectures, where  $M_3=3$ . While, the FFT convolution can be implemented for longer filter kernels of size  $7 \times 7 \times 3$ ,  $15 \times 15 \times 3$ ,  $31 \times 31 \times 3$  up to size that satisfies the application suitability and hardware memory limitation.

**Table 5.1: Fourteen generic 3-D edge and noise smoothing filter operators (kernels), where O.F is Operator Factor**

3-D FIR Filter	3×3×3 kernel		
EdgeXY	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & -1 \\ -1 & 8 & -1 \\ -1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
EdgeX	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
EdgeY	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$
SobelXY	$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -2 & -2 \\ 2 & 0 & -2 \\ 2 & 2 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$
SobelX	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 & -2 \\ 3 & 0 & -3 \\ 2 & 0 & -2 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
SobelY	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 3 & 2 \\ 0 & 0 & 0 \\ -2 & -3 & -2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
Laplacian	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 6 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
PrewittX	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$
PrewittY	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$
Blur O.F= $\frac{1}{6}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Smooth O.F= $\frac{1}{8}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
Sharpen O.F= $\frac{1}{27}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 53 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
Gaussian O.F= $\frac{1}{54}$	$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
Moving Average O.F= $\frac{1}{27}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

### 5.2.3 3-D spatial Convolver Engine

The generalized 3-D spatial convolver engine is a  $(M_1 \times M_2 \times 3)$  MAC FIR digital filter. Thus, this 3-D convolver engine can be realized using three  $(M_1 \times M_2)$  spatiotemporal convolver units and sub-filtered merging stages, as shown in Figure 5.2. Each of the three  $(M_1 \times M_2)$  convolvers consists of  $M_1$  multi-MAC convolver units.

The spatial partitioned block of the 3-D sub-image is sequentially streamed into three  $M_1-1$  row buffers to be filtered in parallel using three  $M_1$  multi-MAC convolver units. Each row buffer effectively delays the input by one row of  $N_1$  pixels, where  $N_1$  is the 3-D image's frame width. An adder tree merges the sub-filtered streams. The resulting pixel values, after applying the image filter, can be negative or larger than 255. Thus, the resulting pixel values are streamed into an absolute unit and Xilinx convert block to take the absolute value from the negative and truncate pixel values larger than 255 to 255 respectively, then, narrow the bit growth to 8 bits.

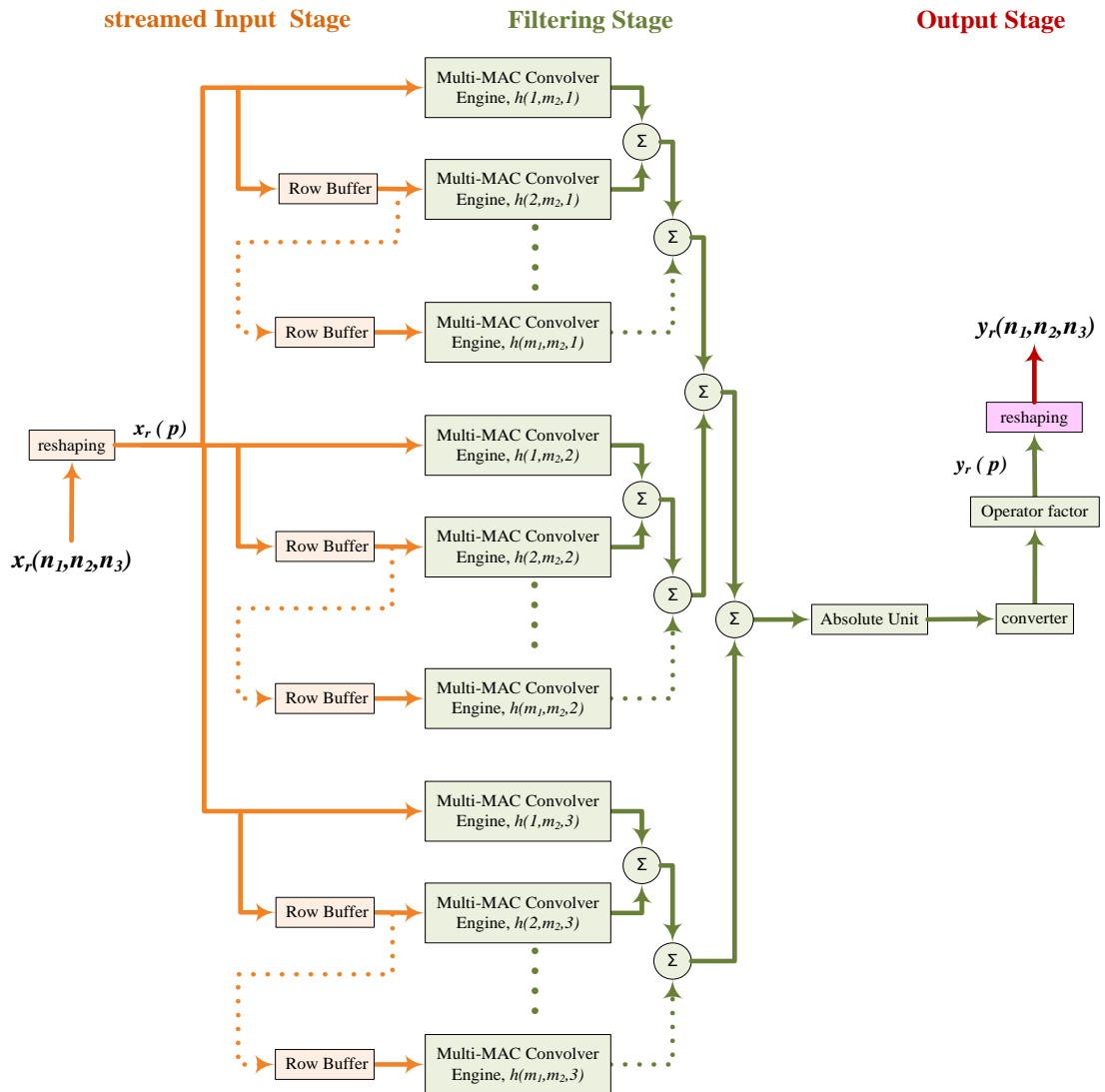


Figure 5.2: Generalized 3-D Spatial Convolver Unit implementation

Consider a  $3 \times 3 \times 3$  window filter, each output filter is a function of the twenty-seven pixel values within the window. Without 3-D stream filtering, twenty-seven pixels must be read, a pixel at each clock cycle, for each window position and each pixel must be read twenty seven times as the window is scanned through the image. A  $3 \times 3 \times 3$  filter kernel spans three image voxels, the current voxel and two previous voxels. Alternatively, the 3-D stream filtering stores the  $3 \times 3 \times 3$  filter kernel into a 3-D MAC FIR engine, and simultaneously stream nine temporal parallelism copies of the 3-D image into nine multi-MAC convolver engines, as generalized in Figure 5.2.

Each three multi-MAC convolver units constitute one of three frames that constitutes the 3-D spatial convolver engine of the 3-D filtering operator. The 3-D spatial convolver is of three-parallelism type: single, dual and quad MAC, as will be explained in subsections 4.45.4.

#### 5.2.4 3-D FFT Convolver Engine

The 3-D spatial convolution, as stated in (5.2), corresponds to complex multiplication of the 3-D FIR filter spectrum and the digital 3-D image spectrum, according to the 3-D circular convolution property of the FFT [39, 41]. This Fourier 3-D image analysis property enables the 3-D FFT convolution to be exploited in linear 3-D image filtering, as shown in Figure 5.3.

The 3-D FFT in the above figure computes the Fourier transform of a 3-D FIR filter  $H(k_1, k_2, k_3)$  and a 3-D digital image  $X(k_1, k_2, k_3)$ , point-by-point multiplied to produce the 3-D filtered image spectrum  $Y(k_1, k_2, k_3)$ . Then, 3-D Inverse FFT computes the spatial filtered 3-D image  $y(n_1, n_2, n_3)$ . Since, the Fourier Transform is inherently separable[19], the computation of the 3-D FFT can be split into the computation of 1-D FFT along rows, columns and frames respectively, as in Figure 5.4. This decomposed 3-D FFT is called row-column-frame FFT (RCFFFT).

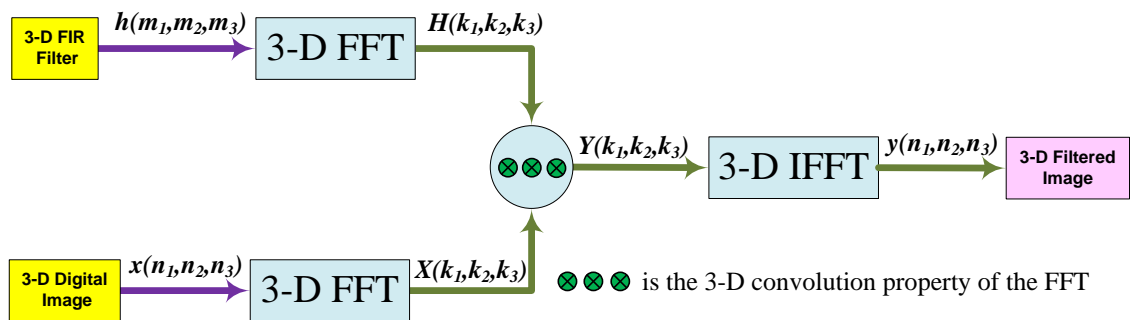
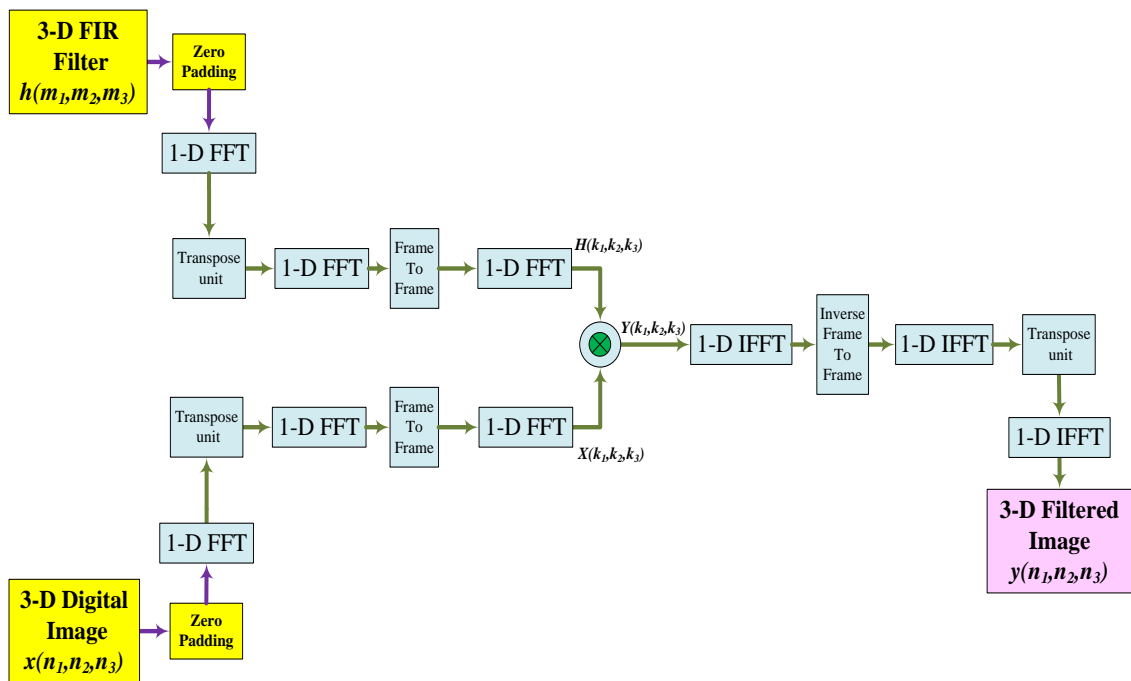


Figure 5.3: Fast filtering by 3-D FFT convolver unit

The Fourier transform is a complex value in the frequency domain, and then the 3-D image will be transformed into an intermediate 3-D image of a real array and an imaginary array. Next, 1-D FFT is repeated along each column of the intermediate 3-D image. Then, the 1-D FFT is computed along each frame. The resulting real and imaginary parts are the 3-D image spectrum,  $X(k_1, k_2, k_3)$ . Similarly, 1-D FFT triplet computed by the 3-D FIR filter spectrum,  $H(k_1, k_2, k_3)$ . These two frequency spectra are point-by-point multiplied to produce the 3-D filtered image spectrum  $Y(k_1, k_2, k_3)$ . To transform back into the spatial domain, the inverse Fourier transform of  $Y(k_1, k_2, k_3)$  is calculated by taking the 1-D inverse FFT along frames, followed by the 1-D inverse FFT long columns, then along the rows respectively.



**Figure 5.4: 3-D separable FFT convolver unit**

The number of complex multiplications and additions required for a radix-2 1-D FFT algorithm of length  $N$  are  $\frac{N}{2} \log_2 N$  and  $N \log_2 N$ , respectively. Thus, the number of complex multiplications and additions needed for the row-column-frame FFT (RCFFFT) that employs such a 1-D FFT are  $\frac{N_1 N_2 N_3}{2} \log_2 N_1 N_2 N_3$  and  $N_1 N_2 N_3 \log_2 N_1 N_2 N_3$ , respectively. Therefore, the RCFFFT reduces the computational complexity from  $O(N^6)$  to  $O(N^3 \log_2 N^3)$ . This considerable gain in computation justifies the hardware implementation of the row-column-frame FFT [19].

### 5.2.5 Parallel 3-D Convolver Architectures Throughput

The total throughput  $\mu$  of the parallel 3-D convolver architectures is measured in a VPS [20, 79, 80]. As a logical extension from the 2-D throughput, the maximum throughput is the maximum operating clock frequency divided by one 3-D volumetric data. However, the throughput of the parallel 3-D spatial convolver architectures is limited by the large input 3-D image and 3-D MAC FIR operation, which can be mitigated by parallelism. Thus, the total throughput is directly proportional to the operating frequency, and inversely proportional to the size of the input 3-D image and the 3-D FIR matrix.

The levels of parallelism are the number  $\alpha_1$  of parallel 3-D spatial convolver filters and the number  $\alpha_2$  refers to the type of MAC engines utilized by the 3-D convolver unit. Thus, the total throughput for the parallel 3-D spatial convolver architectures can be abstracted as:

$$\mu = \frac{f \alpha_1 \alpha_2}{N_1 N_2 N_3 M_1 M_2 M_3} \quad (5.4)$$

where,  $\frac{N_1 N_2 N_3}{\alpha_1} \frac{M_1 M_2 M_3}{\alpha_2}$  is equivalent to one volume time.  $\alpha_1 = 8$  for input 3-D image decimation by 2.  $\alpha_2 = 1, 2$  or 4 using single, dual or quad MAC engine respectively.

For the parallel 3-D FFT convolution architectures, the one volumetric data time is  $(\frac{N_1 N_2 N_3}{\alpha_1} \log_2 \frac{N_1 N_2 N_3}{\alpha_1})$ , since the single 3-D FFT convolution unit performs the 3-D FFT function by  $O(N_1 N_2 N_3 \log_2 N_1 N_2 N_3)$  steps compared to  $O(N_1 N_2 N_3 M_1 M_2 M_3)$  steps of the 3-D spatial convolution [44].

Thus, the overall throughput  $\mu$  of the parallel 3-D FFT convolution architecture of parallel processing stages and maximum clock frequency  $f$  can be formulated as:

$$\mu = \frac{f}{\frac{N_1 N_2 N_3}{\alpha_1} \log_2 \frac{N_1 N_2 N_3}{\alpha_1}} \quad (5.5)$$

### 5.2.6 Total Computation Rate

The total number of MACPS can be considered as another performance index to indicate the computation rate for each one of the five direct architectures. The total computation rate  $\gamma$ , measured in Giga MACPS (GMACPS), is directly proportional to the number of MAC engines and the maximum clock frequency  $f$ . The number of MAC engines is equal to the multiplication of  $\alpha_1, \alpha_2, \alpha_3$  and  $\alpha_4$ , where,  $\alpha_1$  and  $\alpha_2$  as stated in

subsection 5.2.5,  $\alpha_3$  is the number of the parallel spatiotemporal convolver units per 3-D convolver, and,  $\alpha_4$  is the number of multi-MAC convolver per 2-D direct convolver. Then, the computation rate can be formulated as;

$$\gamma = f \times \alpha_1 \times \alpha_2 \times \alpha_3 \times \alpha_4 \quad (5.6)$$

### 5.3 The Generalized Parallel 3-D Linear Image Filtering Algorithm

The generalized parallel 3-D filtering algorithm, as shown in Figure 5.5, for the 3-D image filtering method is presented using the 3-D stream filtering. Generally, this linear 3-D image stream filtering algorithm consists of three stages: 3-D input decimation stage, parallel 3-D sub-filtering processing stage and a parallel 3-D interpolation output stage. The three stages are implemented on the Virtex-6 ML605 development board using XSG. The mathematical model of these three stages for the 3-D stream method is presented in the following sub-sections.

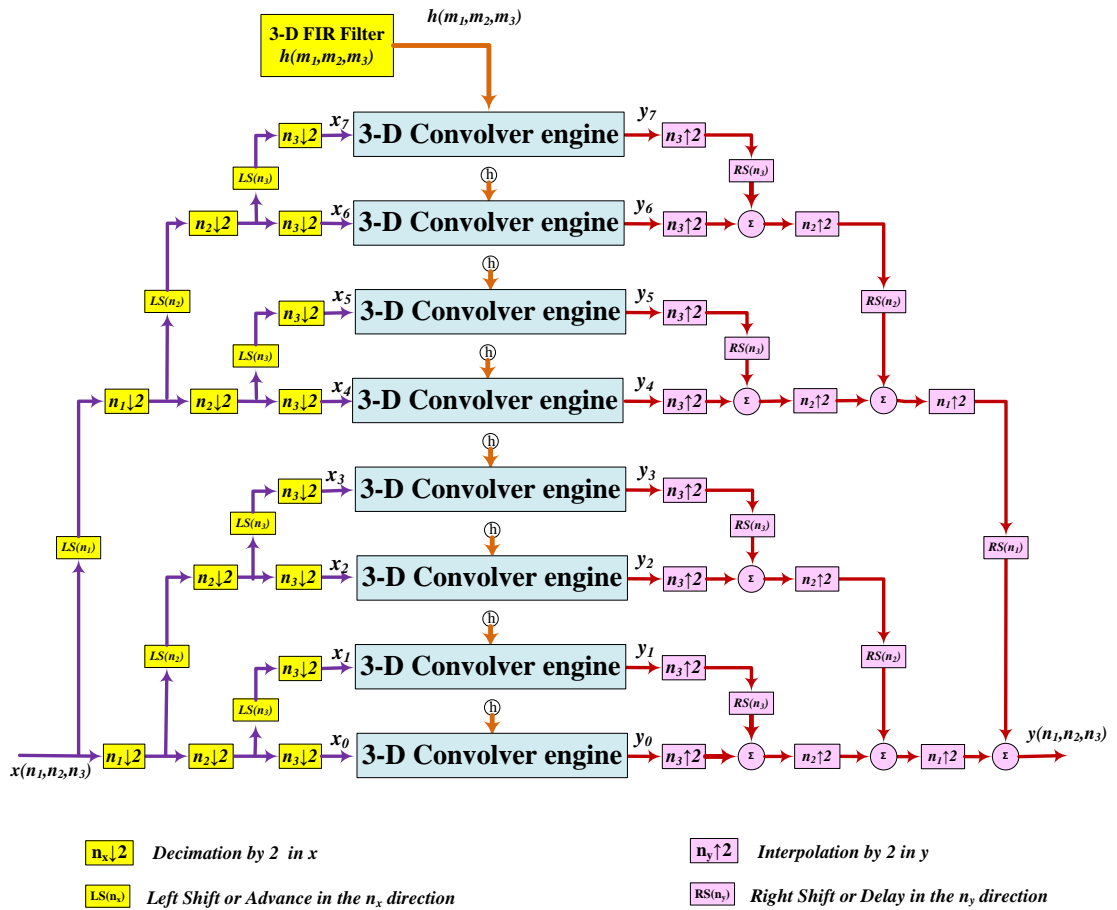


Figure 5.5: The Generalized Parallel 3-D image Linear Filtering Algorithm

### 5.3.1 Input Decimation by 2 for the 3-D Image Stream Filtering

The input digital image  $x(n_1, n_2, n_3)$  of size  $(N_1 \times N_2 \times N_3)$  is decimated by 2 in the three dimensions producing  $(2^3=8)$  sub-image blocks of size  $\left(\frac{N_1}{2}, \frac{N_2}{2}, \frac{N_3}{2}\right)$  for each 3-D image input, as shown in Figure 5.5. These eight 3-D sub-images are not overlapped, and have exactly the same pixels as the 3-D image, hence, the decimation process does not lose or reduce any pixel of the original 3-D image. Thus, these eight 3-D sub-images can be filtered simultaneously and independently. The resultant 3-D sub-image blocks are defined as:

$$\begin{aligned}
 x_0 &= x(2n_1, 2n_2, 2n_3) & x_4 &= x(2n_1+1, 2n_2, 2n_3) \\
 x_1 &= x(2n_1, 2n_2+1, 2n_3) & x_5 &= x(2n_1+1, 2n_2+1, 2n_3) \\
 x_2 &= x(2n_1, 2n_2, 2n_3+1) & x_6 &= x(2n_1+1, 2n_2, 2n_3+1) \\
 x_3 &= x(2n_1, 2n_2+1, 2n_3+1) & x_7 &= x(2n_1+1, 2n_2+1, 2n_3+1)
 \end{aligned} \tag{5.7}$$

The decimated blocks of the input 3-D image are realized by a left shift and decimation in the three dimensions, hence are indicated in Figure 5.5 by LS( $n_1$ )/LS( $n_2$ )/LS( $n_3$ ) and  $n_1 \downarrow 2 / n_2 \downarrow 2 / n_3 \downarrow 2$ , respectively. The resultant down-sampled blocks of the input 3-D image are distinctive and not overlapped, thus, they can be sub-filtered simultaneously and independently in parallel within the 3-D processing stage.

### 5.3.2 Parallel 3-D Filtering Stage

The 3-D processing stage in the parallel 3-D image filtering algorithm is a linear 3-D filtering of the 3-D sub-image given by (5.7) using eight parallel 3-D convolution engines. Each 3-D convolution engine is independently dedicated to one of the eight decimated 3-D sub-images. The linear 3-D filtering can be achieved by either spatial convolution or FFT convolution. This can be expressed by the parallel 3-D convolution equation:

$$y_r(n_1, n_2, n_3) = x_r\left(\frac{n_1}{2}, \frac{n_2}{2}, \frac{n_3}{2}\right) *** h(m_1, m_2, m_3), \quad r=0,1,\dots,7 \tag{5.8}$$

where, \*\*\* is the 3-D convolution operation. The resultant filtering can be expressed as the parallel 3-D convolution equation:

$$y_r(n_1, n_2, n_3) = \sum_{r=0}^7 \left( \sum_{m_1=0}^{\frac{N_1}{2}-1} \sum_{m_2=0}^{\frac{N_2}{2}-1} \sum_{m_3=0}^{\frac{N_3}{2}-1} x_r(m_1, m_2, m_3) h(n_1 - m_1, n_2 - m_2, n_3 - m_3) \right) \tag{5.9}$$



The above equation describes the filtering stage of Figure 5.5, which is computationally intensive. An array of independent parallel 3-D convolution units achieve the filtered 3-D image to speed up the filtering rate, increase the throughput and carry out real time performance for large input 3-D image size.

### 5.3.3 3-D Image Reconstruction Output Stage

The resultant sub-filtered outputs are eight unique 3-D sub-images of  $y_0, y_1, \dots$  and  $y_7$  that are decimated by 2, and they can be represented by the following set of equations:

$$\begin{aligned}
y_0 &= y(2n_1, 2n_2, 2n_3) & y_4 &= y(2n_1 + 1, 2n_2, 2n_3) \\
y_1 &= y(2n_1, 2n_2 + 1, 2n_3) & y_5 &= y(2n_1 + 1, 2n_2 + 1, 2n_3) \\
y_2 &= y(2n_1, 2n_2, 2n_3 + 1) & y_6 &= y(2n_1 + 1, 2n_2, 2n_3 + 1) \\
y_3 &= y(2n_1, 2n_2 + 1, 2n_3 + 1) & y_7 &= y(2n_1 + 1, 2n_2 + 1, 2n_3 + 1)
\end{aligned} \tag{5.10}$$

Thus, the reconstruction of the final output  $y(n_1, n_2, n_3)$  is obtained by two operations: interpolation by 2 and a right shift (or delay), given by  $n_1 \uparrow 2/n_2 \uparrow 2/n_3 \uparrow 2$  and  $RS(n_1)/RS(n_2)/RS(n_3)$  in the three dimensions  $n_1, n_2$  and  $n_3$  respectively, of the eight filtered 3-D sub-images. Which can be mathematically modelled by (5.11), as follows:

$$\begin{aligned}
y(1:2:n_1, 1:2:n_2, 1:2:n_3) &= y_0 & y(2:2:n_1, 1:2:n_2, 1:2:n_3) &= y_4 \\
y(1:2:n_1, 1:2:n_2, 2:2:n_3) &= y_1 & y(2:2:n_1, 1:2:n_2, 2:2:n_3) &= y_5 \\
y(1:2:n_1, 2:2:n_2, 1:2:n_3) &= y_2 & y(2:2:n_1, 2:2:n_2, 1:2:n_3) &= y_6 \\
y(1:2:n_1, 2:2:n_2, 2:2:n_3) &= y_3 & y(2:2:n_1, 2:2:n_2, 2:2:n_3) &= y_7
\end{aligned} \tag{5.11}$$

Where,  $j:i:k$  is the same as  $[j, j+i, j+2i, \dots, k]$ , and represents the two operations of right shift and interpolation by 2.

The generalized parallel 3-D linear stream-filtering algorithm can be realized in hardware architecture (s), as either a 3-D spatial convolution or a 3-D FFT convolution. That depends on the 3-D FIR operator size[44]. The spatial and spectral 3-D image filtering implementations are presented in subsections 5.4 and 5.5 respectively.

## 5.4 Parallel 3-D Spatial Convolver Architectures

The generalized parallel 3-D linear image filtering algorithm of Figure 5.5 can be realized by 3-D spatial convolution units using 3-D MAC FIR filter as shown in Figure 5.6. Consequently, the real-time 3-D image filtering applications in which the 3-D FIR

kernel length shorter than  $(10 \times 10 \times 3)$  coefficients can be achieved using the linear 3-D stream filtering technique, linear 3-D FIR filters and 3-D spatial convolver engines as defined in sub-sections 5.2.1, 5.2.2 and 5.2.3 respectively.

In the input stage, the input 3-D image,  $x(n_1, n_2, n_3)$  of size  $(N_1 \times N_2 \times N_3)$  is decimated by 2 in the three dimensions producing eight 3-D sub-image blocks of size  $(\frac{N_1}{2}, \frac{N_2}{2}, \frac{N_3}{2})$ , designated as  $x_0, x_1 \dots x_7$ , as shown in Figure 5.6.

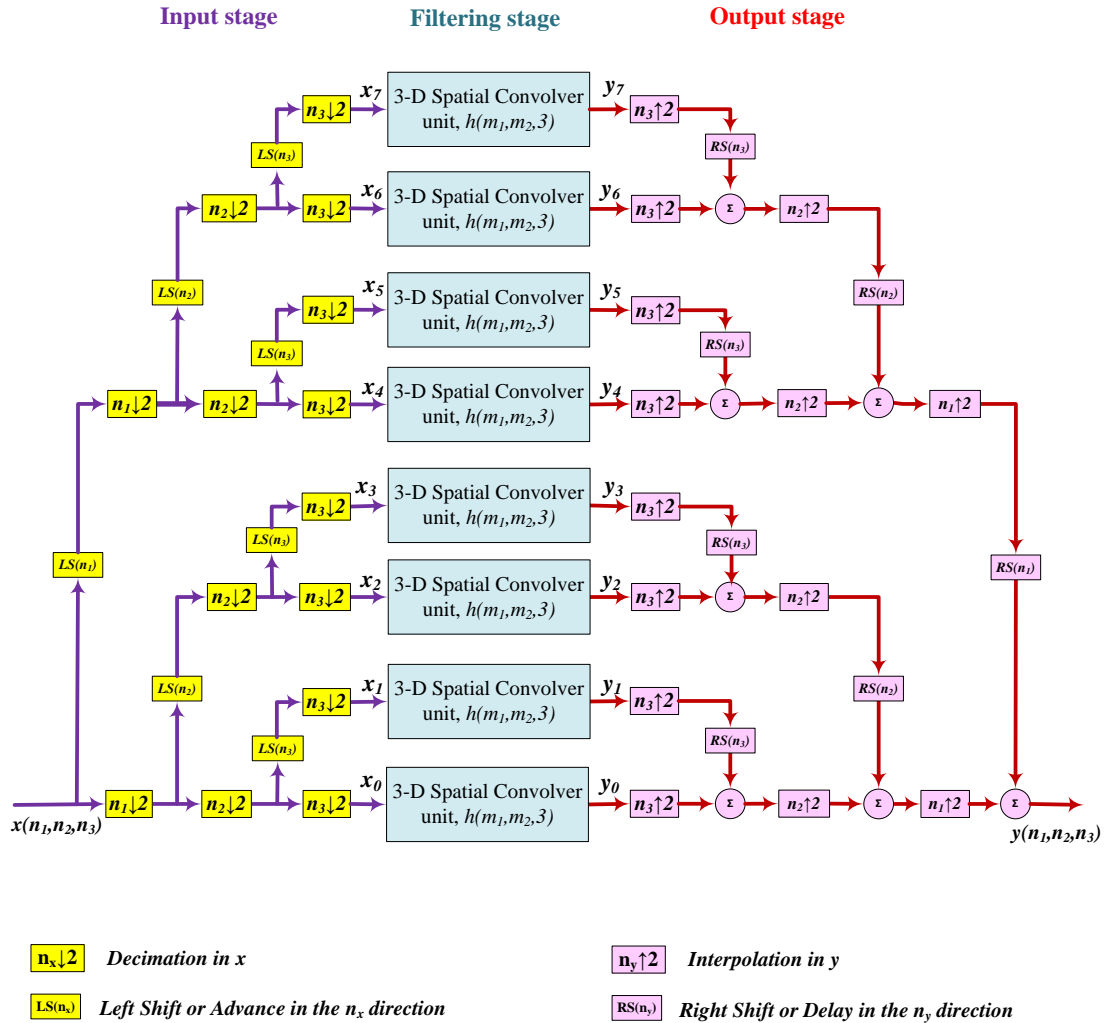
In the processing stage, the parallel filtering operations are performed on-the-fly using eight independent 3-D convolver engines on each decimated block stream. The 3-D convolver engine is a  $(M_1 \times M_2 \times 3)$  MAC FIR digital filter. The filter kernel of  $(M_1 \times M_2 \times 3)$  coefficients is stored in 3-D FIR operator. The spatial partitioned block of the 3-D sub-image is sequentially streamed into  $M_1 - 1$  row buffers to be filtered in parallel using  $M_2$  multi-MAC convolver units, this constitutes one of the three frames of  $(M_1 \times M_2)$  of the 3-D convolver engine. The 3-D convolver engines are of three parallelism types, single MAC, dual MAC and quad MAC, as will be explained in subsection 5.4.1, 5.4.2 and 5.4.3 respectively.

In the output stage as shown in Figure 5.6, the 3-D image  $y(n_1, n_2, n_3)$  is reconstructed out of the eight decimated 3-D sub-images  $y_r(n_1, n_2, n_3)$  by interpolations of  $n_1 \uparrow 2$ ,  $n_2 \uparrow 2$  and  $n_3 \uparrow 2$ , and right shifts or delays of  $RS(n_1)$ ,  $RS(n_2)$  and  $RS(n_3)$ , both in the three dimensions  $n_1$ ,  $n_2$  and  $n_3$  respectively.

The parallel 3-D spatial convolution-filtering algorithm can be implemented by more than one unique FPGA-based architecture according to the MAC FIR hardware structure of the 3-D spatial convolver engine. The MAC FIR hardware structure can be of single MAC, dual MAC or quad MAC. Architectures 16 and 17 use a 3-D single MAC convolver unit, architectures 18 and 19 use a 3-D dual MAC convolver unit. While, the 3-D quad MAC convolver unit is implemented on architecture 20. The input decimated by 2 stage and the output interpolated by 2 stage are the same for all the five spatial architectures.

The decimated blocks of the input 3-D image are realized by left shift and decimation in the three dimensions using the Shift block and Down Sampling block from the XSG blocksets library. The reconstruction of the final output  $y(n_1, n_2, n_3)$  is obtained by two operations: Interpolation by 2 and right shift (or delay) of the eight sub-filtered 3-D

images. These two operations can be realized from the XSG blocksets library by the Up Sampling block and the Shift block or Delay block respectively.

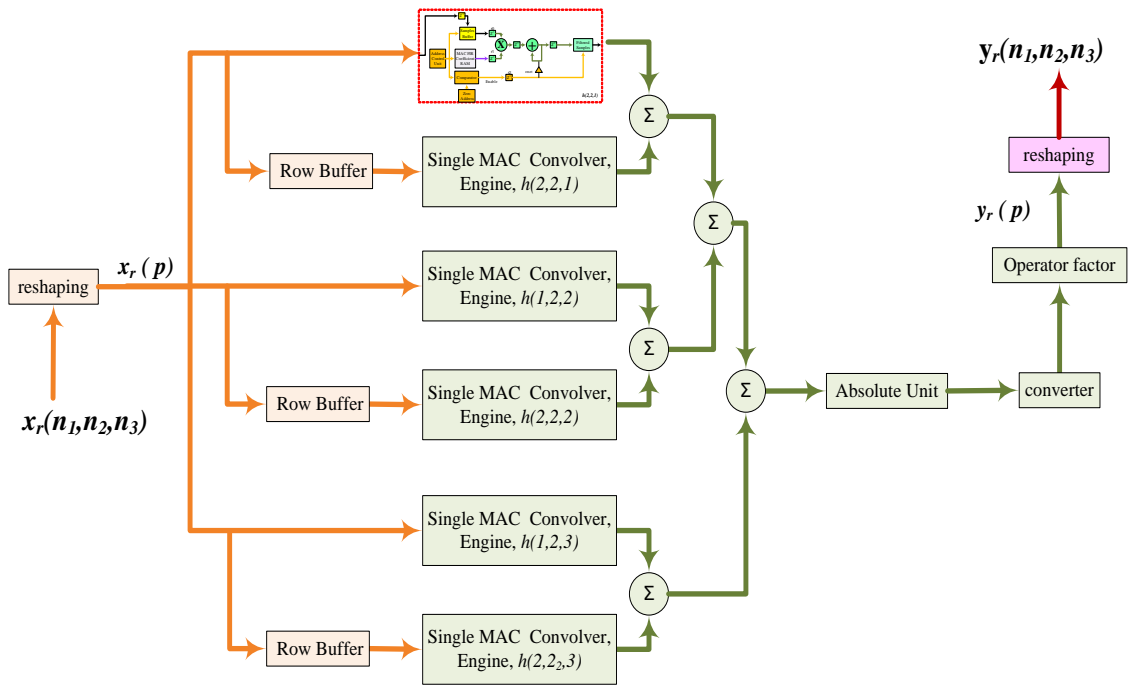


**Figure 5.6: The implementation of the Parallel 3-D Spatial Convolution Algorithm on the Virtex-6 FPGA board; ML 605 development kit**

Therefore, the following architectures can be distinguished by the 3-D MAC engine structure. Thus, the general architecture, shown in Figure 5.6, can have more than one unique hardware version, as explained in the following subsections.

#### 5.4.1 Parallel 3-D Single MAC Convolver Filter Engines Architectures

3-D spatial convolver of the filtering stage, shown in Figure 5.6, can be realized using a 3-D single MAC convolver Units of size  $(2 \times 2 \times 3)$  and  $(3 \times 3 \times 3)$  as parallel architectures 16 and 17 respectively. Figure 5.7 and Figure 5.8 shows  $(2 \times 2 \times 3)$  and  $(3 \times 3 \times 3)$  as unit 7 and unit 8 respectively.

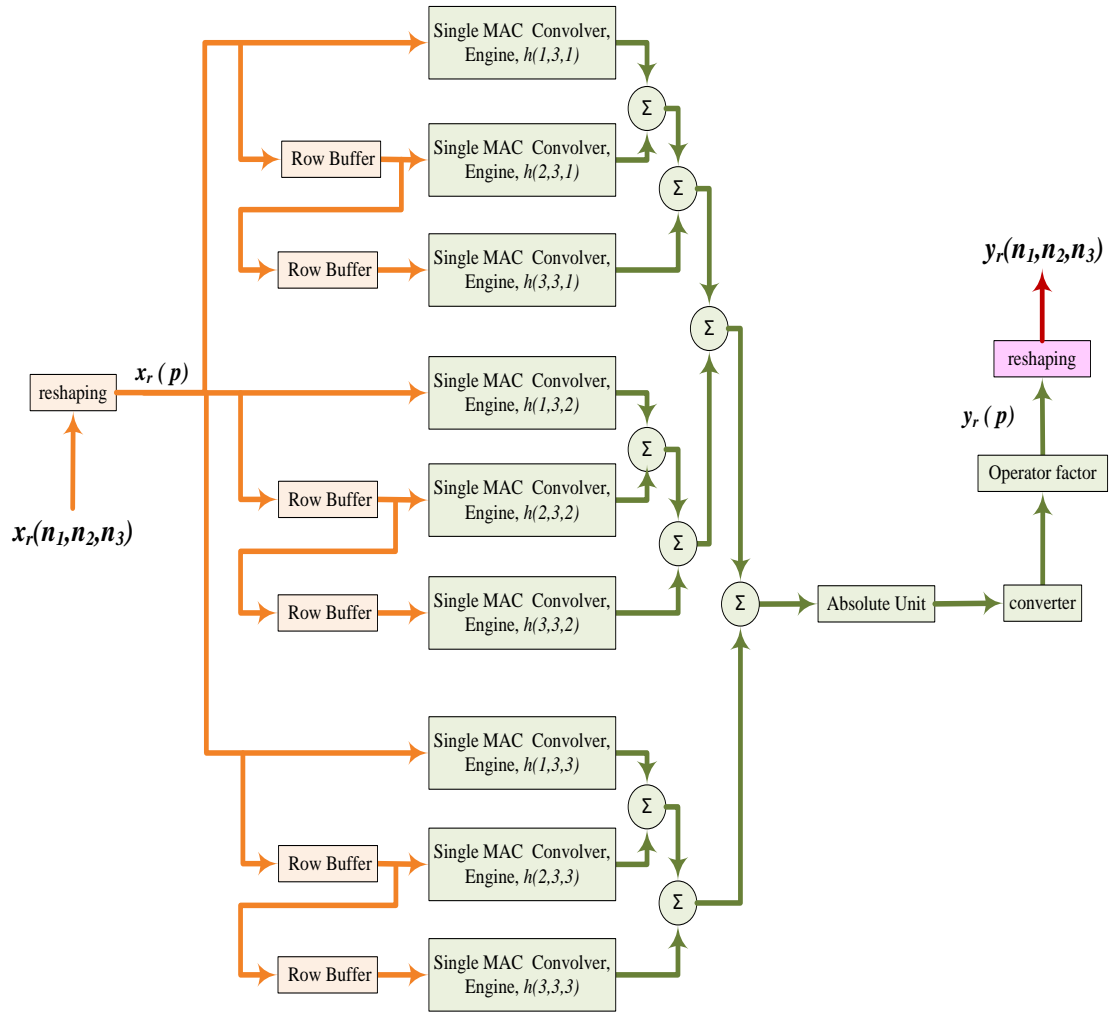


**Figure 5.7: Unit 7; (2×2×3) Single MAC convolver Architecture**

Thus, the (3×3×3) single MAC engine consists of three of unit 8, as implemented on Appendix A. The logic area occupied by architecture 17 is mapped in Appendix B using Xilinx FPGA editor Tool. The schematic RTL diagram is graphically depicted in Appendix C. As configured and shown in Table 5.2, unit 7 occupies less logic area than unit 8 by (39%) FFs, (44%) LUTs, (40%) slices, (33%) dedicated DSP 48E1s multiplier, and (40%) RAMB 18E1s block memory.

**Table 5.2: Logic Devices utilization by the 3-D spatial convolver units**

	3-D Single MAC convolver units		3-D Dual MAC convolver units		3-D Quad MAC convolver unit
	Unit 7 2×2×3	Unit 8 3×3×3	Unit 9 2×4×3	Unit 10 4×4×3	Unit 11 2×8×3
<b>FFs</b>	477	775	686	1 326	1 022
<b>LUTs</b>	296	526	491	988	807
<b>Slices</b>	139	228	200	384	272
<b>DSP 48E1s</b>	6	9	12	24	24
<b>RAMB 18E1s</b>	9	15	15	33	27



**Figure 5.8: Unit 8; (3×3×3) Single MAC convolver Architecture**

#### 5.4.2 Parallel 3-D Dual MAC Convolver Filter Engines Architectures

The 3-D spatial convolver unit, shown in Figure 5.6, can be realized using 3-D dual MAC operator Units of size  $(2 \times 4 \times 3)$  and  $(4 \times 4 \times 3)$  as in architectures 18 and 19. Figure 5.9 and Figure 5.10 shows  $(2 \times 4 \times 3)$  and  $(4 \times 3 \times 3)$  as unit 9 and unit 10 respectively.

As configured and shown in Table 5.2, unit 9 occupies less logic area than unit 10 by (48%) FFs, (50%) LUTs, (48%) slices, (50%) the dedicated DSP 48E1s multiplier, and (45%) the RAMB 18E1s block memory. By comparing the logic area occupation of the single MAC and dual MAC units' architectures, the highest logic area occupation is in unit 10 and descending to unit 8, unit 9 down to the lowest in unit 7.

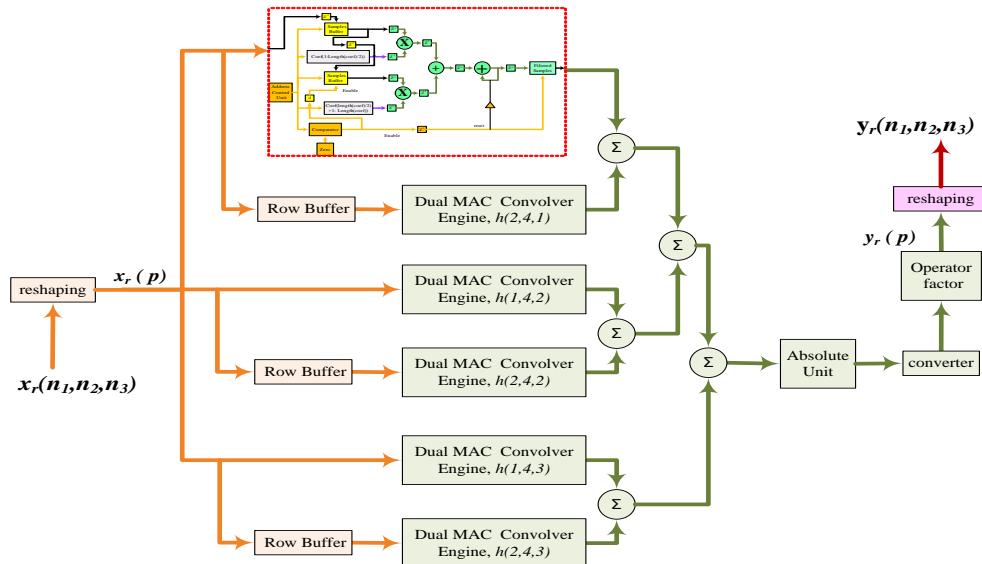


Figure 5.9: Unit 9; (2×4×3) Dual MAC convolver Architecture

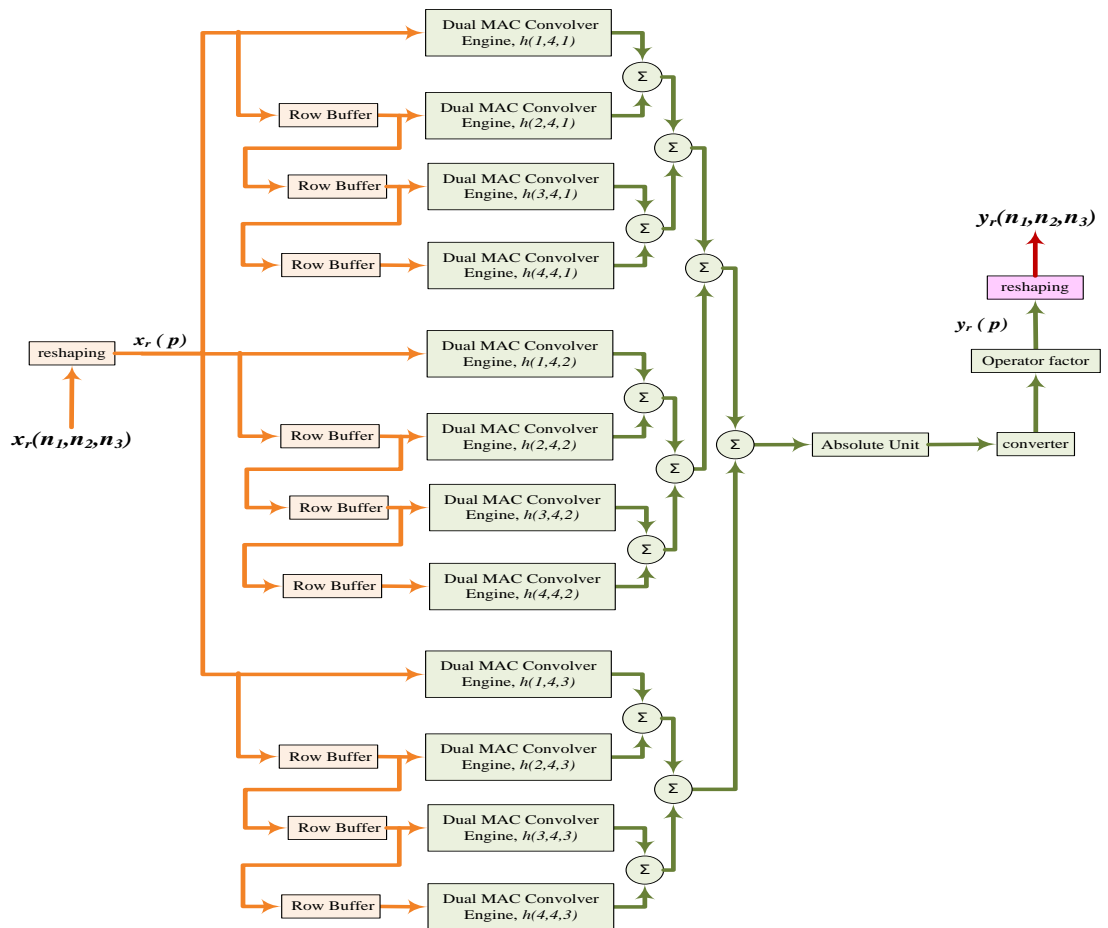
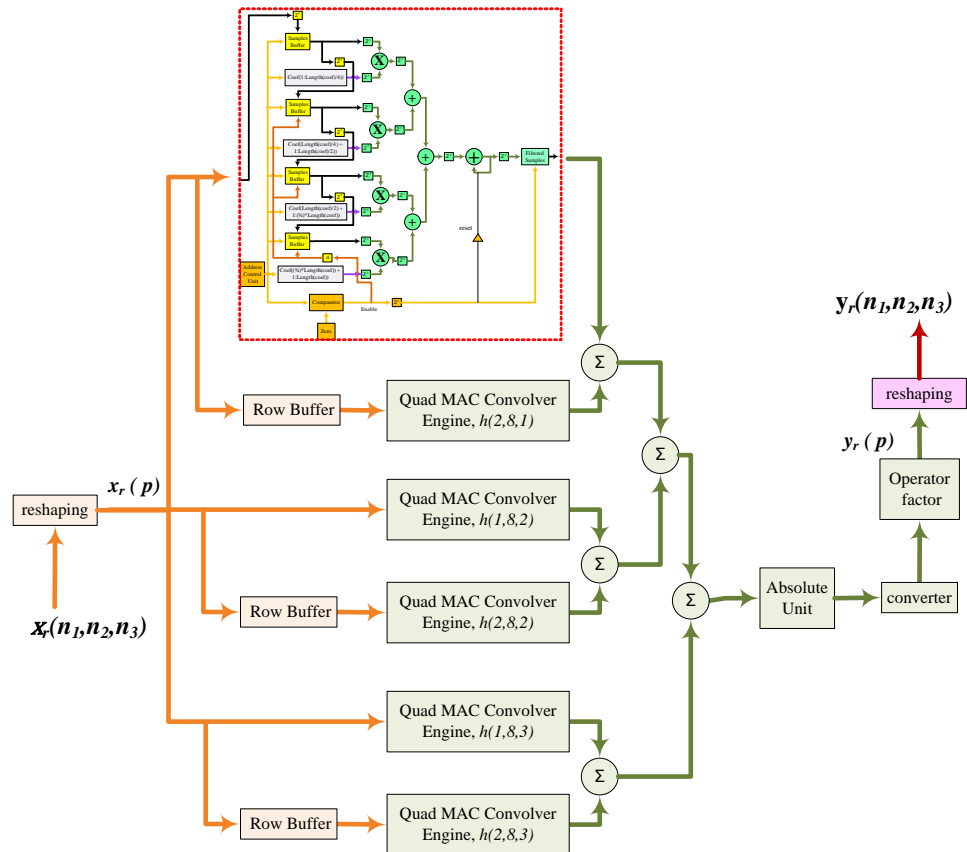


Figure 5.10: Unit 10; (4×4×3) Dual MAC convolver Architecture

#### 5.4.3 Parallel (2×8×3) Quad MAC 2-D Convolver Filter Engine Architecture

Architecture 20 is distinguished by using (2×8×3) quad MAC 3-D direct convolver engine, unit 11, to implement its 3-D FIR filter operator, as shown in Figure 5.11.

As configured and shown in Table 5.2, unit 11 occupies less logic area than unit 10 by (23%) FFs, (18%) LUTs, (29%) slices, with the same number of dedicated DSP 48E1s multipliers, and (18%) less RAMB 18E1s block memory. Moreover by comparing the logic area occupation of the single MAC, dual MAC and quad MAC unit architectures, the logic area occupation ascending from the lowest in unit 7, unit 9, unit 8, unit 11 and up to highest in unit 10.



**Figure 5.11: Unit 11; (2x8x3) Quad MAC convolver Architecture**

#### 5.4.4 Performance Indices of Parallel 3-D Spatial Convolver Architectures

The performance indices of the five parallel spatial convolver architectures are considered as a complete set of area, speed, power, throughput and computation rate using XSG to target a Virtex-6 ML605 board. The minimized utilized area of the five architectures, as shown in Table 5.3, are due to the efficient implementation hierarchy of logic fabric, signals, I/O's and hard IPs such as Block RAMs or DSP blocks, and occupying proper resources of FFs, LUTs and slices.

The highest logic area utilization is in Architecture 19, occupying only (3%) FFs, (4%) LUTs, (10%) Slices, (25%) dedicated DSP 48E1s and (31%) RAMB 18E1s block memory of the available logic assets of the targeted Virtex-6 XC6VLX240T development FPGA board, while the lowest logic area utilization of architecture 16

occupies one third the logic area and one quarter the Hard IPs of architecture 19. Moreover by comparing the logic area occupation of the 3-D single, dual and quad MAC architectures, the logic area occupation ascending from the lowest in architectures 16, 18, 17, 20 and up to the highest in architecture 19. Consequently, this area occupation affects the performance indices set of power consumption, throughput and computation rate as shown in Table 5.4. Where, the speed is the maximum clock frequency, the power consumption is the dynamic power, the total throughput and computation rate are both affected by the degree of parallelism, that is, the internal parallelism style of the 3-D spatial convolver engines as well as the overall architecture's parallelism, as defined in subsections 5.2.5 and 5.6 respectively.

**Table 5.3: Logic Devices utilization by each of the five parallel spatial convolver architectures**

	Archit. 16	Archit. 17	Archit. 18	Archit. 19	Archit. 20
	<b>3-D FIR filter kernel (<math>M_1 \times M_2 \times M_3</math>)</b>				
	$2 \times 2 \times 3$	$3 \times 3 \times 3$	$2 \times 4 \times 3$	$4 \times 4 \times 3$	$2 \times 8 \times 3$
<b>FFs</b>	3 992	7 464	5 424	10 520	8 112
<b>LUTs</b>	2 625	5 390	3 825	7 641	6 392
<b>Slices</b>	1 082	1 972	1 494	3 421	2 404
<b>DSP 48E1s</b>	48	72	96	192	192
<b>RAMB 18E1s</b>	72	120	120	264	216

Several observations can be made from Table 5.4. Firstly, the operating clock frequency is particularly insensitive to the occupied logic area by the five architectures because of pipelining, and principally operating around the indicated 200 MHz maximum frequency [31]. Secondly, the dynamic power consumption at (40 nm) junction temperature of 54°C decreases, monotonically, from 259 mW, 212 mW, 120 mW, 97 mW down to 64mW for architectures 19, 20, 18, 17 and 16 respectively. Thus, the power consumption improved more than 4-fold. Thirdly, the highest throughput is achieved by architecture 16, 18 and 20 almost (2.25) and (4) times that of architecture 17 and 19 respectively, that is, according to (5.4), where, the throughput of the parallel 3-D spatial convolver architectures is limited by the large input 3-D image and 3-D MAC FIR operation. Thus for the same 3-D FIR filter kernels, the throughput for the greyscale 256×256×20 MRI filtering is higher than that for greyscale 1024×1024×20 MRI by more than 16-fold. Fourthly, according to (5.6), the highest computation rate is accomplished by architecture 19 and 20, due to the four levels of parallelism. That is double, more than (2.5) and triple than that of architectures 18, 17 and 16 respectively.

**Table 5.4: Performance indices of each of the 3-D MAC FIR filter architectures**



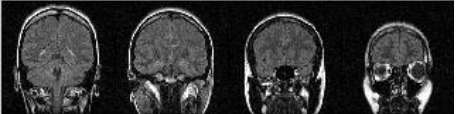

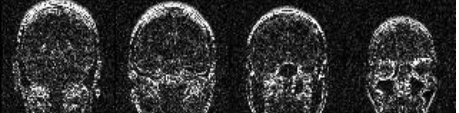

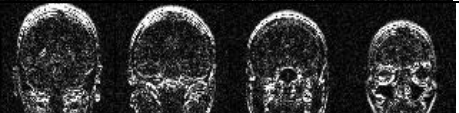





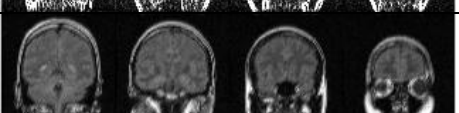
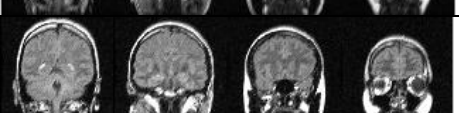
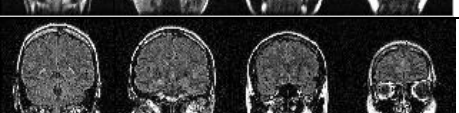
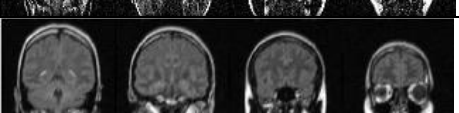
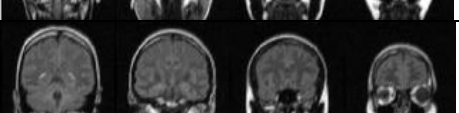
	Archit. 16	Archit. 17	Archit. 18	Archit. 19	Archit. 20
3-D FIR filter kernel ( $M_1 \times M_2 \times M_3$ )					
	2×2×3	3×3×3	2×4×3	4×4×3	2×8×3
<b>Maximum Clock Frequency (MHz)</b>	205	203	205	201	205
<b>Dynamic Power (mWatt)</b>	64	97	120	259	212
<b>Throughput (VPS) greyscale 256×256×20 MRI</b>	104	46	104	51	104
<b>Throughput (VPS) greyscale 1024×1024×20 MRI</b>	7	3	7	3	7
<b>computation rate (GMACPS)</b>	10.8	16.2	21.6	43.2	43.2

The five implementations of the parallel 3-D spatial convolution algorithm are designed as “plug and develop” architectures. Thus, fourteen generic ( $3 \times 3 \times 3$ ) FIR filters are plugged into 3-D edge detection and 3-D noise smoothing circuits, then, improved their suitability for this particular biomedical imaging application of greyscale  $256 \times 256 \times 20$  volume MRI, as shown in Table 5.5 and Table 5.6 respectively using Architecture 17.

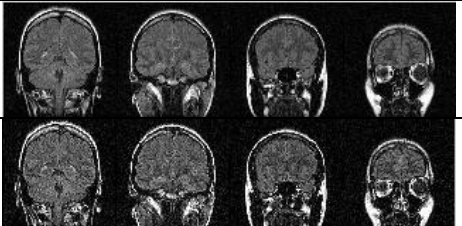
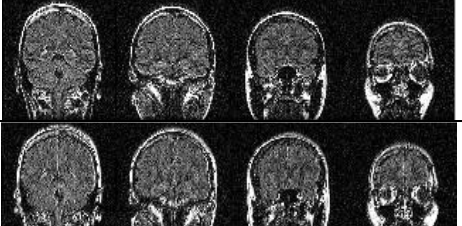
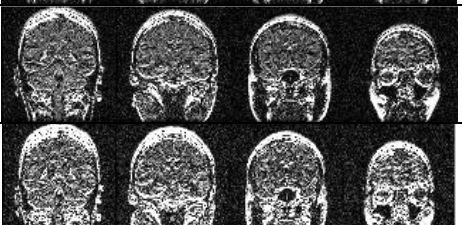
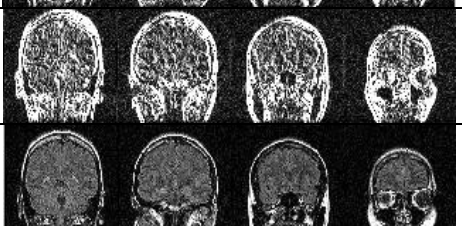
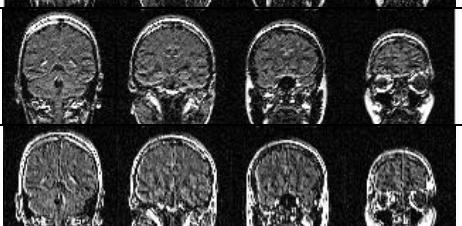
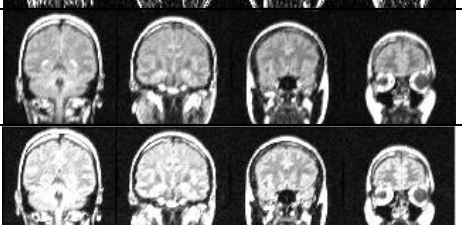
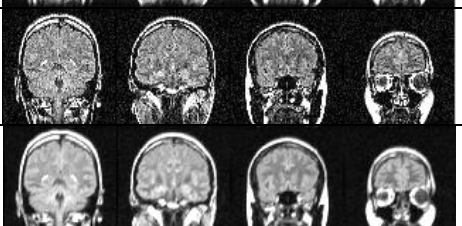
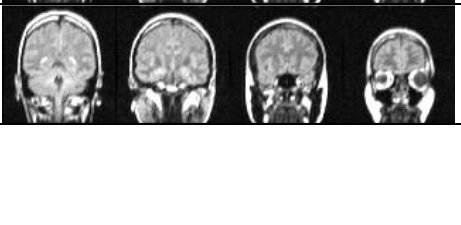

The 3-D MRI filtered images of Table 5.5 show fourteen generic ( $3 \times 3 \times 3$ ) filtering operators, nine edge and five noise smoothing. The 3-D Edge operators detect the changes or differences in voxel value at the edges of volume. Thus, the 3-D Edge operators are differencing filters, which make them sensitive to noise, as in the first nine operators of Table 5.5, which act as low pass filters except the Laplacian filter. This high pass filter can be used to detect edges of all orientations. The five noise smoothing operators prevent the output value being different from the input in uniform regions of the 3-D MRI volume.

Table 5.6 shows the improvement in each of the 3-D edge filtering operators by designing the corresponding Edge enhancement operator. While, the five noise smoothing operators are heuristically improved by incrementing the coefficients operator’s value to be 2. The 3-D filtering result is to be the same brightness as the original MRI, but sharper.

**Table 5.5: The filtering results of greyscale 256×256×20 volume fMRI for fourteen generic 3-D FIR filter operators using architecture 17**

3-D FIR Filter	3×3×3 kernel			
	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & -1 \\ -1 & 8 & -1 \\ -1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>EdgeXY</b>	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & -1 \\ -1 & 8 & -1 \\ -1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>EdgeX</b>	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>EdgeY</b>	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	
<b>SobelXY</b>	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & -2 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
<b>SobelX</b>	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>SobelY</b>	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	
<b>Laplacian</b>	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 6 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>PrewittX</b>	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>PrewittY</b>	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	
<b>Blur</b> O.F= $\frac{1}{6}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>Smooth</b> O.F= $\frac{1}{12}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
<b>Sharpen</b> O.F= $\frac{1}{27}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 53 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
<b>Gaussian</b> O.F= $\frac{1}{56}$	$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
<b>Moving Average</b> O.F= $\frac{1}{27}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

**Table 5.6: The filtering results for grayscale  $256 \times 256 \times 20$  volume fMRI of fourteen improved 2-D FIR filter operators using architecture 17**

3-D FIR Filter	3×3×3 kernel			
EdgeXY O.F= $\frac{1}{8}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & -1 \\ -1 & 16 & -1 \\ -1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
EdgeX O.F= $\frac{1}{2}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 4 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	
EdgeY O.F= $\frac{1}{2}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 4 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	
SobelXY	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & -2 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
SobelX	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	
SobelY	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	
Laplacian O.F= $\frac{1}{6}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 12 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
PrewittX O.F= $\frac{1}{3}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 2 \\ -1 & 0 & 2 \\ -1 & 0 & 2 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	
PrewittY O.F= $\frac{1}{3}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & 2 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	
Blur O.F= $\frac{1}{6}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 2 & 1 & 2 \\ 2 & 0 & 2 \\ 2 & 1 & 2 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Smooth O.F= $\frac{1}{12}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 8 & 2 \\ 2 & 2 & 2 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
Sharpen O.F= $\frac{1}{27}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 70 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Gaussian O.F= $\frac{1}{56}$	$\begin{bmatrix} 2 & 4 & 2 \\ 4 & 0 & 4 \\ 2 & 4 & 2 \end{bmatrix}$	$\begin{bmatrix} 4 & 8 & 4 \\ 8 & 16 & 8 \\ 4 & 8 & 4 \end{bmatrix}$	$\begin{bmatrix} 2 & 4 & 2 \\ 4 & 0 & 4 \\ 2 & 4 & 2 \end{bmatrix}$	
Moving Average O.F= $\frac{1}{27}$	$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$	

### 5.5 Four Dimension (4-D) Application: fMRI or 4-D Colour MRI Volume Filtering

Colour 4-D MRI volume is a multispectral image where each voxel is represented by a quad-tuple. Thus, the 4-D colour MRI volume is a 4-D matrix of size  $(N_1 \times N_2 \times N_3 \times N_4)$ . Where, the fourth dimension takes the values of 0, 1 and 2 for the red, green and blue MRI components respectively. Then, fMRI or 4-D colour MRI is represented as a vector-base image  $(n_1, n_2, n_3)$  of  $(N_1 \times N_2 \times N_3 \times 3)$  size,

$$\mathbb{x}(n_1, n_2, n_3) = \begin{bmatrix} x_1(n_1, n_2, n_3) \\ x_2(n_1, n_2, n_3) \\ x_3(n_1, n_2, n_3) \end{bmatrix} \quad (5.12)$$

where,  $x_1(n_1, n_2, n_3)$ ,  $x_2(n_1, n_2, n_3)$  and  $x_3(n_1, n_2, n_3)$  are the MRI components of red, green and blue of size  $(N_1 \times N_2 \times N_3)$  each. Colour 4-D MRI or fMRI is typically represented by a four-dimensional vector of eight bits per component, resulting in a 32-bit colour system. 4-D colour MRI filtering involves the 3-D convolution of each component (red, green and blue) of  $\mathbb{x}(n_1, n_2, n_3)$  with a 3-D FIR operator  $h(m_1, m_2, m_3)$  of  $(M_1 \times M_2 \times 3)$  size to produce the three component filtered output  $\mathbb{y}(n_1, n_2, n_3)$ , given by,

$$\begin{bmatrix} y_1(n_1, n_2, n_3) \\ y_2(n_1, n_2, n_3) \\ y_3(n_1, n_2, n_3) \end{bmatrix} = \begin{bmatrix} x_1(n_1, n_2, n_3) \\ x_2(n_1, n_2, n_3) \\ x_3(n_1, n_2, n_3) \end{bmatrix} *** h(m_1, m_2, m_3) \quad (5.13)$$

where, \*\*\* is the 3-D convolution, which can be expressed in details as:










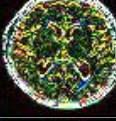










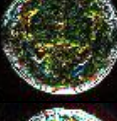
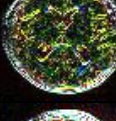

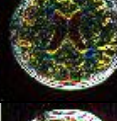








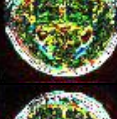







$$y(n_1, n_2, n_3) = \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} \sum_{m_3=0}^{N_3-1} x(m_1, m_2, m_3) h(n_1 - m_1, n_2 - m_2, n_3 - m_3) \quad (5.14)$$

The real-time 4-D colour MRI volume filtering of the above equation can be implemented using the 3-D implementations of architectures 16, 17, 18, 19 and 20, due to their quicker filtering speed, high computation rate and higher throughput at low dynamic power consumption and small logic area occupation, as shown in Table 5.3 and Table 5.4. Thus, the performance indices of the five architectures for the colour 4-D MRI volume filtering are the same as those indices for the greyscale 3-D MRI volume. However, the total throughput is less by a factor of three times, as shown in Table 5.7, because of the three frames of red, green and blue that constitutes each colour MRI slice of the fMRI. Using architecture 17 as a four dimensions colour image reconfigurable processor to “plug and filter” the 4-D colour MRIs or fMRIs are shown in Table 5.8 and Table 5.9. Then, these fourteen 3-D MRI filtering operators are used to improve their suitability for this particular biomedical imaging application, as shown in Table 5.10 and Table 5.11. All the filtering results tables are obtained using Architecture 17.

**Table 5.7: Throughput of fMRI (colour MRI) volume using architectures 16 to 20**


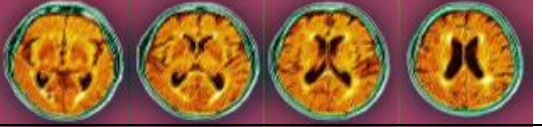
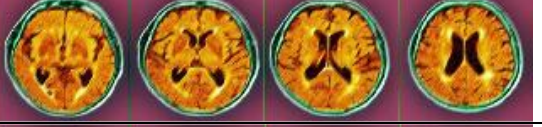



	Archit.	Archit.	Archit.	Archit.	Archit.
	16	17	18	19	20
	3-D FIR filter kernel ( $M_1 \times M_2 \times M_3$ )				
	2×2×3	3×3×3	2×4×3	4×4×3	2×8×3
<b>Throughput (VPS) colour 256×256×20×3 MRI</b>	35	15	35	17	35
<b>Throughput (VPS) colour 1024×1024×20×3 MRI</b>	2	1	2	1	2

**Table 5.8: Filtering results for colour 256×256×3×4 MRI volumetric**

3-D FIR Filter	Generic 3×3×3 kernel				
<b>EdgeXY</b>	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & -1 \\ -1 & 8 & -1 \\ -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$				
<b>EdgeX</b>	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$				
<b>EdgeY</b>	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$				
<b>SobelXY</b>	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & -2 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$				
<b>SobelX</b>	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$				
<b>SobelY</b>	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$				
<b>Laplacian</b>	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 6 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$				
<b>PrewittX</b>	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$				
<b>PrewittY</b>	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$				













**Table 5.9 : the 4-D filtering results for colour 256×256×3×4 MRI volume of five generic 3-D FIR noise smoothing filter operators using architecture 17**







3-D FIR Filter	Generic 3×3×3 kernel	
<b>Blur</b> O.F= $\frac{1}{6}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>Smooth</b> O.F= $\frac{1}{12}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
<b>Sharpen</b> O.F= $\frac{1}{27}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 53 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
<b>Gaussian</b> O.F= $\frac{1}{56}$	$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
<b>Moving Average</b> O.F= $\frac{1}{27}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

The improvement in each of the 3-D edge-filtering operators is achieved by designing the corresponding 3-D Edge enhancement operators. The 3-D Edge enhancement works as a sharpening operation by boosting the high frequency content of the 4-D MRI. While, the five 3-D noise smoothing operators are heuristically improved for the colour 4-D MRI suitability by incrementing the operator's coefficients sum value to be 2. The 4-D filtering result is to be the same brightness as the original MRI, but sharper.

**Table 5.10: the 4-D filtering results for colour 256×256×3×20 MRI volume of nine improved 3-D FIR Edge enhancement filter operators using architecture 17**

2-D FIR Filter	Developed 3×3×3 kernel	
EdgeXY O.F= $\frac{1}{8}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & -1 \\ -1 & 16 & -1 \\ -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
EdgeX O.F= $\frac{1}{2}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 4 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	
EdgeY O.F= $\frac{1}{2}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 4 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	
SobelXY	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & -2 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
SobelX	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & -2 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	
SobelY	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	
Laplacian O.F= $\frac{1}{6}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 12 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
PrewittX O.F= $\frac{1}{3}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 2 \\ -1 & 0 & 2 \\ -1 & 0 & 2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	
PrewittY O.F= $\frac{1}{3}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 2 & 2 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	







**Table 5.11: the 4-D filtering results for colour  $256 \times 256 \times 3 \times 20$  MRI volume of five improved 3-D FIR noise smoothing filter operators using architecture 17**

2-D FIR Filter	Developed $3 \times 3 \times 3$ kernel	
<b>Blur</b> $O.F = \frac{1}{6}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 & 2 \\ 2 & 0 & 2 \\ 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
<b>Smooth</b> $O.F = \frac{1}{12}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 2 & 2 \\ 2 & 8 & 2 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
<b>Sharpen</b> $O.F = \frac{1}{27}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 70 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
<b>Gaussian</b> $O.F = \frac{1}{56}$	$\begin{bmatrix} 2 & 4 & 2 \\ 4 & 0 & 4 \\ 2 & 4 & 2 \end{bmatrix} \begin{bmatrix} 4 & 8 & 4 \\ 8 & 16 & 8 \\ 4 & 8 & 4 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 4 & 0 & 4 \\ 2 & 4 & 2 \end{bmatrix}$	
<b>Moving Average</b> $O.F = \frac{1}{27}$	$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$	

As a comparison of the filtering results, the five architectures are used, as shown in Table 5.12, to process the same colour 4-D MRI volume using the  $(M_1 \times M_2 \times M_3)$  3-D FIR operators of architectures 18, 19, 20, 21 and 22. The filtered 4-D MRI volume are identically 4-D edge enhanced and brightened due to the sum of the  $(M_1 \times M_2 \times M_3)$  elements equal to 2. Therefore, these parallel 3-D spatial convolution architectures can be utilized as colour image reconfigurable processors acting as open development multidimensional filtering engines.



**Table 5.12: Filtering results for 256×256×3×20 colour MRI of the improved 3-D Edge Enhancement operators using architectures 16, 17, 18, 19 and 20.**

3-D Operator ( $M_1 \times M_2 \times 3$ )	
$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$	
$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	
$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	

The 4-D filtering results within the above tables show the 4-D edge filtering, 4-D noise smoothing and 4-D edge enhancement for diverse regions of a 76-year-old patient with brain dementia. The filtered fMRI or colour 4-D MRI volume can be used by the physician to noninvasively depict areas of the brain for investigation prior to neurosurgery and the brain functional activities in detail that are used for specific tasks [98, 104, 113, 114, 129].

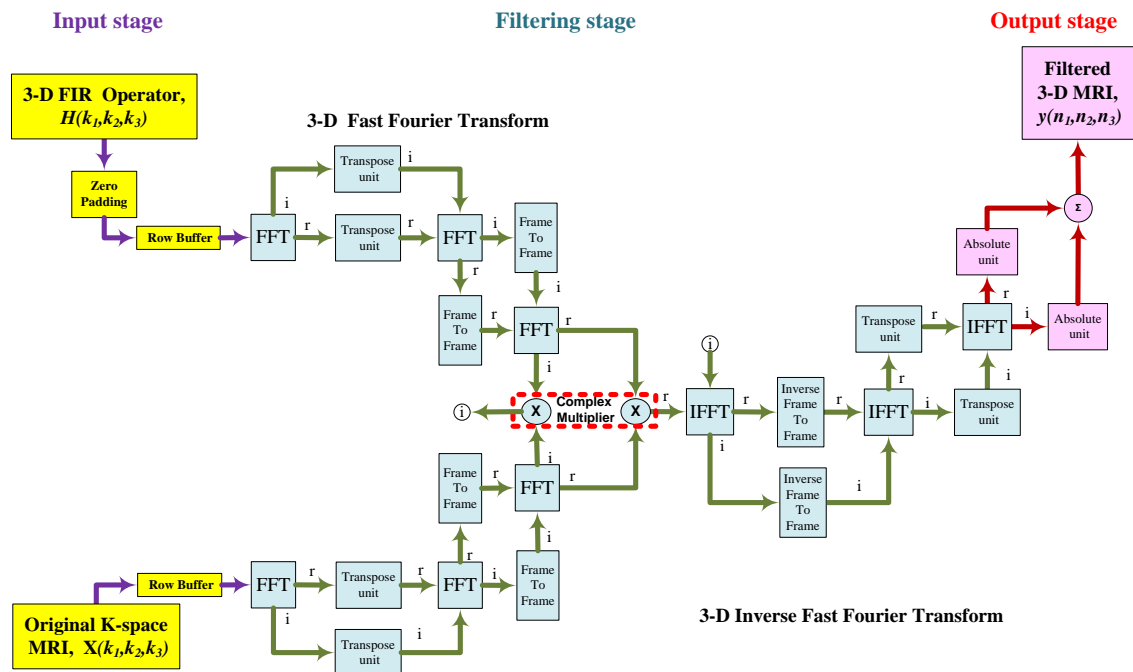
### 5.6 FPGA Implementation as Parallel 3-D FFT Convolver Architectures

The parallel 3-D FFT convolution-filtering algorithms can be applied in the 3-D image filtering applications in which the 3-D FIR kernel length is longer than (10×10×3) coefficients, as had been shown in subsections 5.2.4 and 5.2.5. This 3-D FFT algorithms' FPGA implementation can be efficiently realized by more than one of the fast parallel 3-D FFT filtering architectures, depending on the 3-D image segmentation, FPGA memory and accordingly the parallelism of the filtering stage. Consequently, two efficient hardware architectures are developed; single 3-D FFT convolver architecture and parallel 3-D FFT convolver architecture. The first approach is described in subsection 5.6.1 as architecture 21, and in subsection 5.6.2 as a new application for a

real-time k-space application as architecture 22. The second approach is described in subsection 5.6.3 as architecture 23.

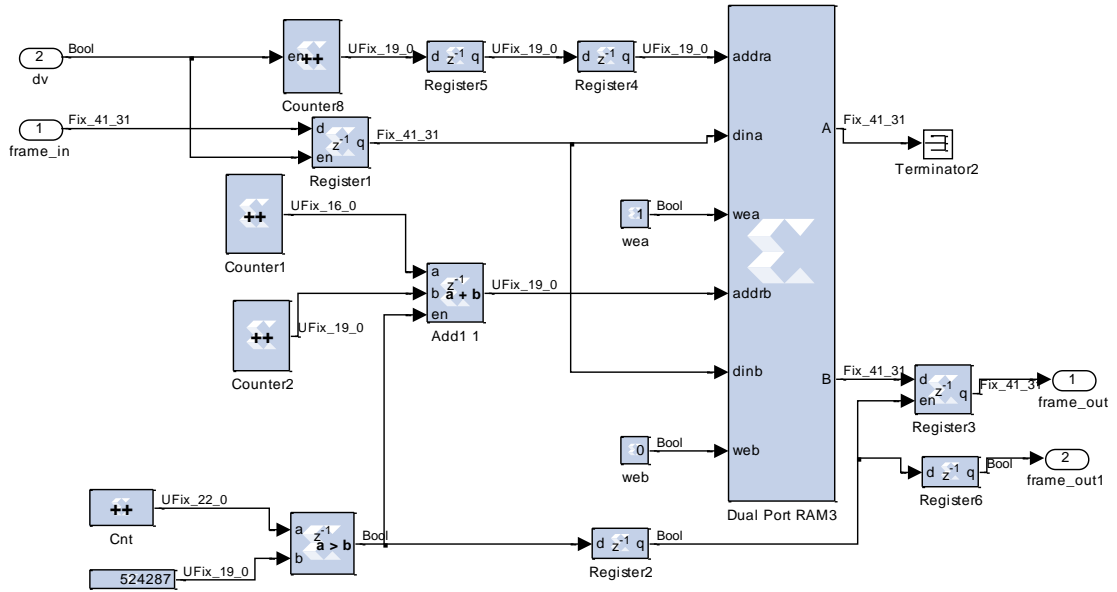
### 5.6.1 Single 3-D FFT Convolver Unit Implementation

The 3-D FFT convolver engine unit of Figure 5.5 can be realized as a single 3-D FFT convolver architecture as shown in Figure 5.12. This architecture consists of input, filtering and output stages. Within the input stage, the 3-D MRI and the 3-D FIR operators are streamed using the linear 3-D Stream Filtering Method, as explained in subsection 5.2.1. The temporal stream of the 3-D FIR operator is zero padded to be equal in size to the 3-D MRI. Both streams are row buffered to be processed in the filtering stage. The row buffer component is implemented as shown in Figure 4.12.



**Figure 5.12: Architecture 21; the implementation of the Fast Single 3-D FFT convolution unit in the Virtex-6 FPGA board**

The 3-D filtering stage is a spatial to frequency transformation structure, as explained in subsection 5.2.4. This frequency transformation structure consists of a 3-D FFT, a complex multiplier and a 3-D IFFT. The 3-D FFT consists of three Xilinx FFT v7\_0 blocks, two transpose unit and two frame-to-frame structures. The transpose units are realized as shown in Figure 4.18, and explained in subsection 4.6.1. The frame-to-frame units can be realized as shown in Figure 5.13, where, the 2-D frequency transformed 3-D volume is first stored via port A of a dual port RAM, and then accessed in port B by two address control circuitry.



**Figure 5.13: frame-to-frame unit implementation**

The writing address circuitry is an up counter from zero to  $(N_1 \times N_2 \times N_3 - 1)$  to store the 3-D image stream row by row. While the reading address circuitry is a combination of two up counters to generate the 3-D volume out of port B row by row for a transformed frame by using a frame-to-frame transformation algorithm, as shown in following pseudo-code,

```

% frame to frame transformation algorithm
R=row; C=column; F=frame;
Mri_volume=1:R*C*F;
z1=1;
% frame to frame transformation
for j=1:R*C
  for i=j:R*C:R*C*F
    transformed_mri_volume (z1,1)=mri_volume(i,1);
    z1=z1+1;
  end
end

```

The two 3-D matrices of the FIR coefficients and the MRI are simultaneously frequency transformed via the real inputs of three Xilinx FFT v7\_0 blocks, and setting the imaginary input to zero. Each FFT outputs a frequency spectrum of real and imaginary parts. Thus, the image will be transformed into an intermediate image of real array and imaginary array. Subsequently, two transpose units are used, then, 1-D FFT is repeated on each column of the intermediate image. Next, two frame-to-frame units are used so that the third dimension is transformed to the frequency domain. The resulting two parts of real and imaginary are the 3-D image frequency spectrum,  $X(k_1, k_2, k_3)$ . Similarly, 1-D

FFT triplet compute the 3-D FIR operator frequency spectrum,  $H(k_1, k_2, k_3)$ . Using Xilinx complex multiplier block, these two frequency spectra are point-by-point multiplied to produce  $Y(k_1, k_2, k_3)$ , which represents the 3-D FFT of the filtered image. To transform back into the spatial domain, the inverse Fourier transform of  $Y(k_1, k_2, k_3)$  is computed by taking the 1-D inverse FFT of each transformed frame. Then, two inverse frame-to-frame units are used to transform back the volume to the previous frame-to-frame format, followed by two transpose units to arrange the two filtered image components, then the third 1-D inverse FFT of each row is processed. The inverse frame-to-frame transformation algorithm is illustrated by the following pseudo-code,

**% inverse frame to frame transformation algorithm**

```

R=row; C=column; F=frame;
Mri_volume=1:R*C*F;
z1=1;
for j=1:F
  for i=j: F: R*C*F
    inverse_transf._mri_volume (z1,1)= transformed_mri_volume(i,1);
    z1=z1+1;
  end
end

```

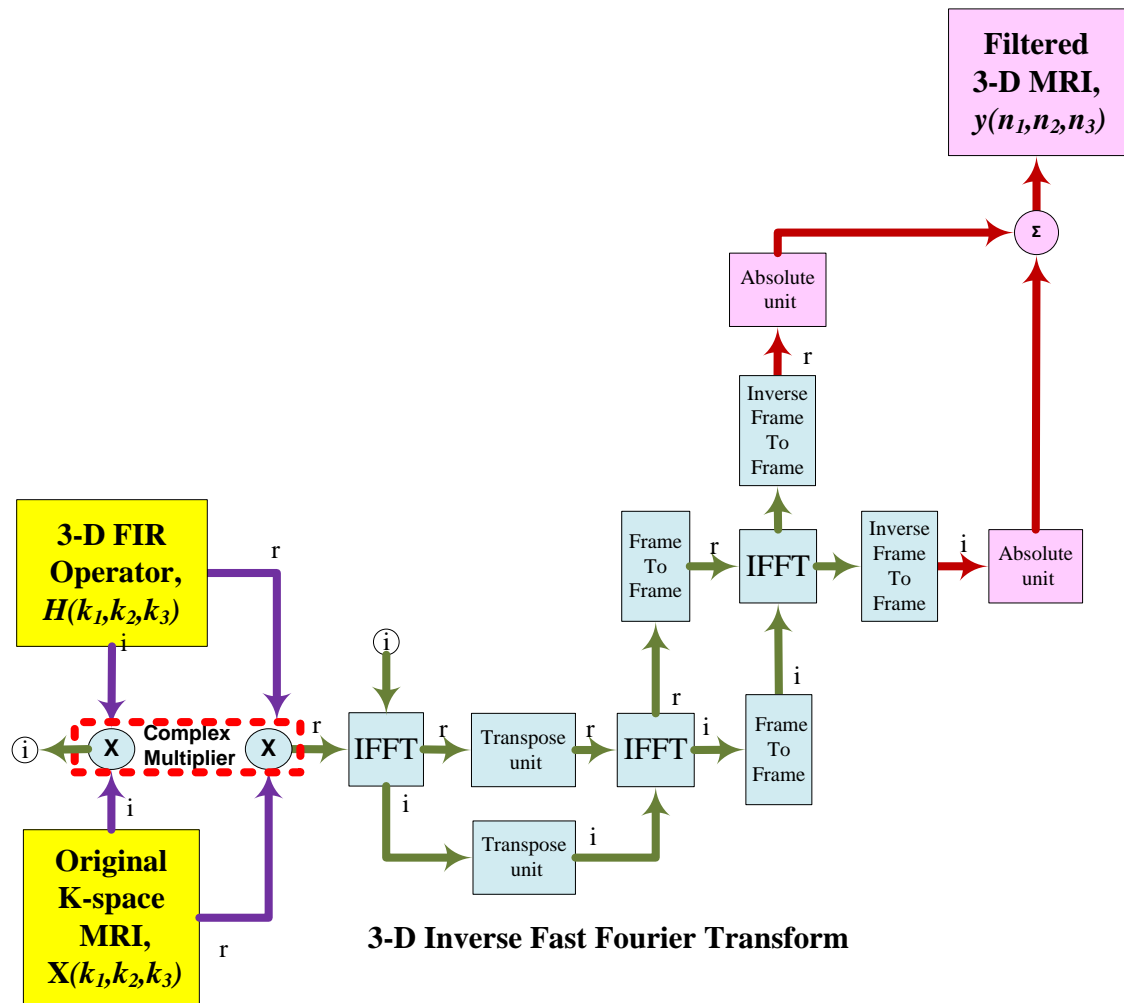
In the output stage, the two streams of the filtered MRI are conditioned, first, by the absolute unit, and then summed to produce the 3-D filtered MRI  $y(n_1, n_2, n_3)$ . The final reconstructed output is connected to a gateway-out block, which provides the conversion from the fixed point format which is used by the FPGA to floating point numerical representation used by Simulink blocks for displaying the filtered MRI.

**5.6.2 Single 3-D IFFT Convolver k-Space MRI Filtering Unit Implementation**

The 3-D MRI data volume or fMRI are collected from the MRI machine in the frequency volume distributed within a k-space [116-118]. Noise and artifacts for various sources often corrupt these frequency volume data. Thus, the filtering of this raw frequency data can vastly improve 3-D MRI visualization. This filtering can efficiently be achieved using a single 3-D IFFT convolution algorithm. An FPGA implementation of a single 3-D IFFT convolver-filtering unit is shown in Figure 5.14.

Where, the two frequency inputs; the original k-space 3-D MRI data  $X(k_1, k_2, k_3)$  and the 3-D FIR operator kernel  $H(k_1, k_2, k_3)$  are point-by-point complex multiplied to produce  $Y(k_1, k_2, k_3)$  as the 3-D FFT of the filtered 3-D image. Then, the resultant real and imaginary 3-D filtered MRI has to be transformed back into the spatial domain by a 3-D

IFFT. The inverse Fourier transform of  $Y(k_1, k_2, k_3)$  is computed by taking the 1-D inverse FFT row by row. Then, two transpose units are used to precede the second 1-D IFFT, followed by two frame-to-frame units to prepare the third dimension to be transformed back by the third 1-D inverse FFT. The final filtering results are obtained by inverse frame to frame units to transform the 3-D MRI volume to the original row-column-frame format.



**Figure 5.14: Architecture 22; an implementation of Fast Single 3-D IFFT convolution k-space MRI volume data Filtering unit in the Virtex-6 FPGA board**

In the output stage, the two streams of the filtered 3-D MRI are conditioned, first, by an absolute unit, and then summed to produce the 3-D filtered MRI  $y(n_1, n_2, n_3)$ . The final reconstructed output is connected to a gateway-out block, which provides the conversion from the fixed-point format, used by the FPGA, to floating-point numerical representation used by Simulink blocks for displaying the filtered 3-D MRI.

### 5.6.3 Fast Parallel 3-D FFT Convolution Architecture

Currently, most of the 3-D image processors are limited by the memory restriction to handle the huge 3-D image volume data [40, 42, 112, 123, 128, 130]. To resolve this hardware limitation necessitates that the 3-D input image is segmented into independent 3-D sub-images. Consequently, a parallel fast convolution algorithm has to be devised to simultaneously process these 3-D sub-images. Thus, the generalized parallel 3-D filtering algorithm of Figure 5.5 may be realized in hardware as architecture 23, shown in Figure 5.15. This architecture consists of input, filtering and output stages. The input decimated by 2 stage and the output interpolated by 2 stage are as described in subsection 5.3.1 and 5.3.3 respectively. The main difference is in the filtering stage which in this case is based on the convolution property of the 3-D FFT.

The input 3-D MRI is decimated by 2 to produce eight independent sub-MRI blocks  $x_0, x_1 \dots x_7$ . These 3-D sub-images and the 3-D FIR operator are converted from spatial parallelism to temporal streams as described in subsection 5.2.1. Each stream is row buffered before being processed within the filtering stage. Due to the independency of the eight decimated sub-streams, the fast 3-D filtering stage is carried out simultaneously using parallel 3-D FFT convolvers array. Consequently, there are no internal communications in the convolution-filtering array due to boundary conditions as with the block filtering method [41].

The filtering stage consists of eight parallel 3-D Fast Fourier Transforms (FFT) convolution structure. Each 3-D FFT convolution structure is a fast single 3-D FFT convolution sub-architecture, as described in subsection 5.6.1, where the spatial 3-D filtering convolution is achieved by parallel complex multiplication. The eight sub-MRI pixels streams and the 3-D FIR operator are frequency transformed by 3-D FFT, complex multiplied and spatially transformed by 3-D inverse FFT. The resultant filtered sub-MRI has two components of real and imaginary. Each component is conditioned by an absolute unit and summed to produce eight decimated by 2 filtered sub-mages of  $y_0, y_1 \dots y_8$ , as shown in Figure 5.15. The final filtered 3-D MRI is reconstructed from the eight 3-D sub-MRI using interpolation by 2. This output signal is connected to a gateway-out block for the fixed-point to floating-point conversion of the filtered 3-D MRI so that it can be by displaying via Simulink blocks.

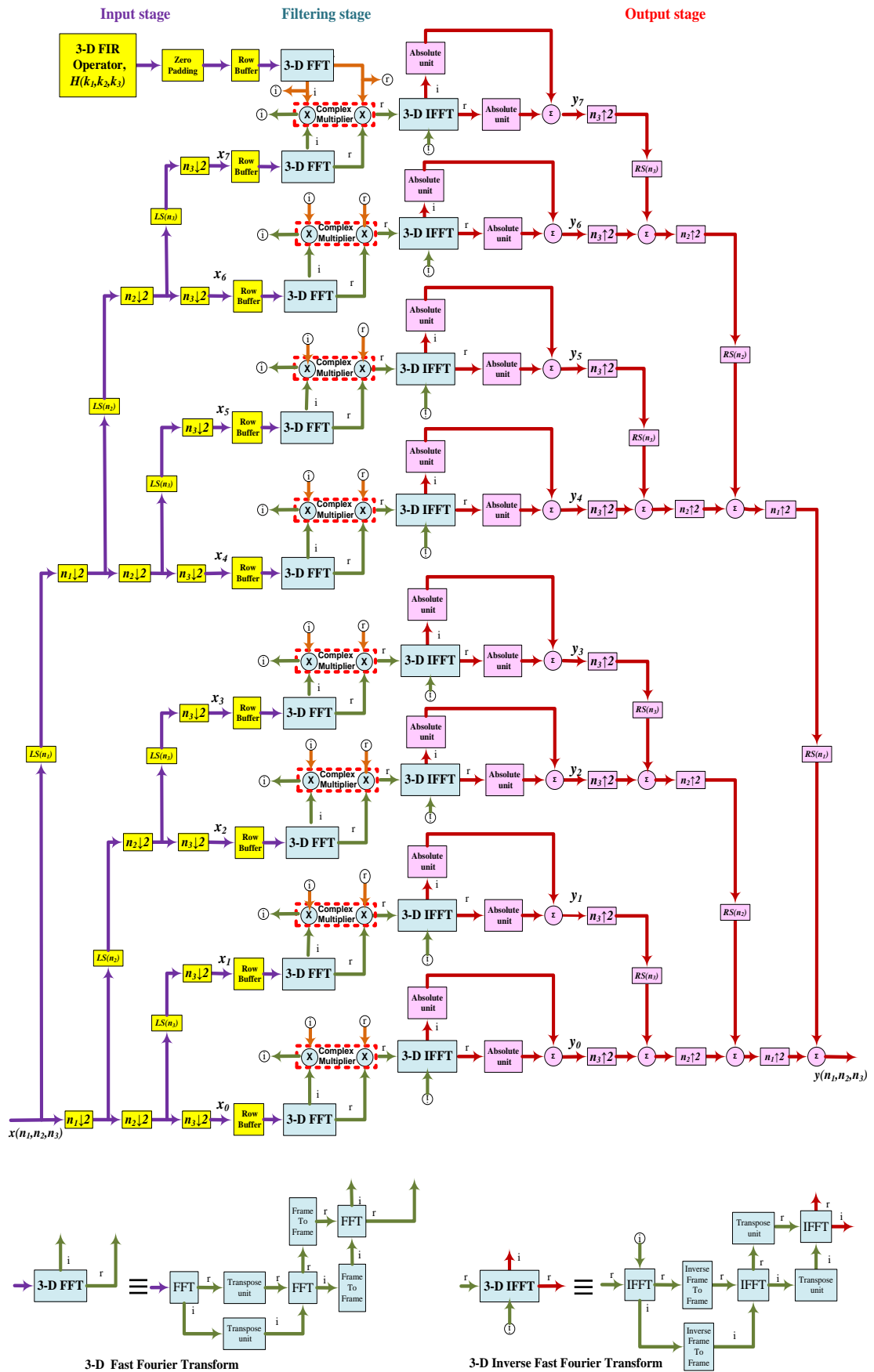


Figure 5.15: Architecture 23; an implementation of Fast Parallel 3-D FFT convolution Algorithm in the Virtex-6 FPGA board

#### 5.6.4 Performance Indices of Parallel 3-D FFT Convolver Architectures

The performance indices of three 3-D FFT convolver architectures are considered as a complete set of area, speed, power and throughput using XSG to target a Virtex-6 ML605 board. The results are presented into logic devices utilization tables, performance indices tables and 3-D MRI volume data filtered image tables.

The main variables that affected the 3-D image filtering results are the 3-D image size, 3-D FIR operator size and accordingly the size of N-point FFT/IFFT triplet. Hence, comparative evaluation results for the three FPGA implementation architectures are obtained for grayscale 3-D MRI volume data of  $(32 \times 32 \times 8)$  and  $64 \times 64 \times 8$  size and distinctive 3-D FIR operator of  $(7 \times 7 \times 3)$ ,  $(15 \times 15 \times 3)$  and  $(31 \times 31 \times 3)$  coefficients, to be individually applied with a (16, 32 and 64) N-point FFT/IFFT triplet. Accordingly, the area occupation, performance indices and 3-D MRI filtered outputs of the two architectures are shown in Table 5.13, Table 5.14, Table 5.19, Table 5.16, Table 5.17 and Table 5.20 respectively. These two architectures occupy proper logic area of FFs, LUTs, slices, DSP blocks and Block RAMs, so that, architecture 22 was in average occupying less logic fabric than one-third and half DSP blocks/BRAMs than that architecture 21. Moreover, the area occupation, performance indices and 3-D MRI filtered outputs of architecture 23 are presented in Table 5.15, Table 5.18 and Table 5.21 respectively.

**Table 5.13: Logic Devices utilization by architecture 21**

Greyscale MRI size ( $N_1 \times N_2 \times N_3$ )	3-D FIR kernel ( $M_1 \times M_2 \times M_3$ )	FFT N-point	FFs	LUTs	Slices	DSP 48E1s	RAMB 18E1s
32×32×8	15×15×3	32	12733	7347	2529	120	75
64×64×8	31×31×3	64	14437	8414	3090	126	116

**Table 5.14: Logic Devices utilization by architecture 22**

Greyscale MRI size ( $N_1 \times N_2 \times N_3$ )	3-D FIR kernel ( $M_1 \times M_2 \times M_3$ )	FFT N-point	FFs	LUTs	Slices	DSP 48E1s	RAMB 18E1s
32×32×8	15×15×3	32	9323	5534	1690	67	36
64×64×8	31×31×3	64	10417	6557	2269	67	64

**Table 5.15: Logic Devices utilization by architecture 23**

Greyscale MRI size ( $N_1 \times N_2 \times N_3$ )	3-D FIR kernel ( $M_1 \times M_2 \times M_3$ )	FFT N-point	FFs	LUTs	Slices	DSP 48E1s	RAMB 18E1s
32×32×8	7×7×3	16	94249	55913	20371	544	616



The performance indices set of speed, power consumption and throughput as shown in Table 5.16 and Table 5.17. Several observations can be made from the tables. Firstly, the maximum clock frequency is steadily increasing [33] as the 3D MRI data volume size, 3-D operator size and FFT N-point triplet are decreasing. Secondly, the power consumption is monotonically decreases as the 3-D MRI volume data size, 3-D operator size and FFT triplet N-point are decreasing. Thirdly, architecture 22 outperformed architecture 21 in maximum clock frequency and power consumption. Fourthly, architecture 22's throughput outperformed that of architecture 21, in average, by two-folds within all the corresponding 3-D MRI volume data size, 3-D operator size and FFT triplet N-point.

**Table 5.16: Performance indices of architecture 21**

Grayscale MRI size ( $N_1 \times N_2 \times N_3$ )	3-D FIR kernel ( $M_1 \times M_2 \times M_3$ )	FFT N-point length	Maximum clock frequency (MHz)	Dynamic Power (mWatt)	Throughput (VPS)
32×32×8	15×15×3	32	233	243	2 188
64×64×8	31×31×3	64	214	496	435

**Table 5.17: Performance indices of architecture 22**

Grayscale MRI size ( $N_1 \times N_2 \times N_3$ )	3-D FIR kernel ( $M_1 \times M_2 \times M_3$ )	FFT N-point length	Maximum clock frequency (MHz)	Dynamic Power (mWatt)	Throughput (VPS)
32×32×8	15×15×3	32	248	112	4 647
64×64×8	31×31×3	64	221	249	899

**Table 5.18: Performance indices of architecture 23**

Grayscale MRI size ( $N_1 \times N_2 \times N_3$ )	3-D FIR kernel ( $M_1 \times M_2 \times M_3$ )	FFT N-point length	Maximum clock frequency (MHz)	Dynamic Power (mWatt)	Throughput (VPS)
32×32×8	7×7×3	16	211	842	20 605

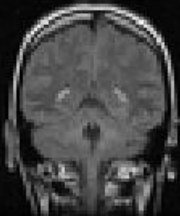
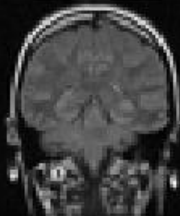
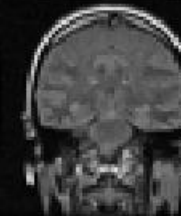
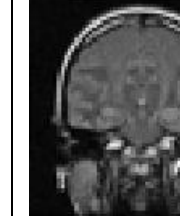

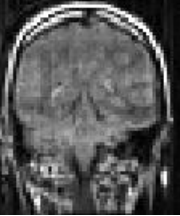
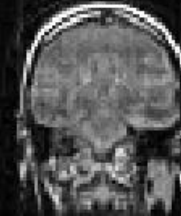

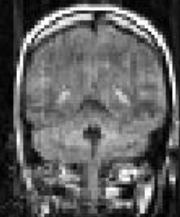
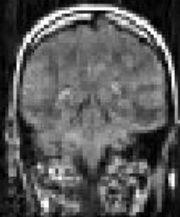


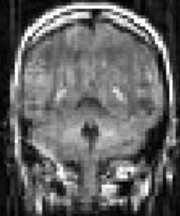
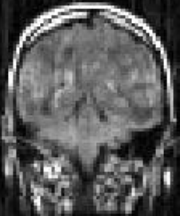
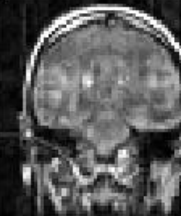

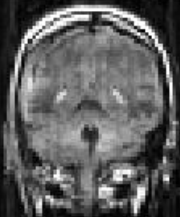
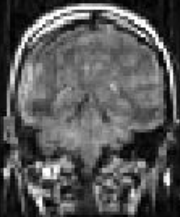

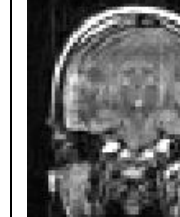
Architecture 23 is the parallel version to filter 3-D MRI data volume using architecture 21 as the 3-D convolver engine. Thus, architecture 21 occupied less logic area than architecture 25 by (13%) FFs and LUTs, (12%) slices, (22%) dedicated DSP 48E1s multiplier, and (13%) RAMB 18E1s block memory. Consequently, architecture 23 was consuming dynamic power of 842 mW to produce more than nine-folds increase in filtering throughput at a maximum clock frequency of 211 MHz. Thus, the parallelisms in the algorithm side and the implementation resulted in architecture 23 to have the highest throughput.

In three dimensions MRI filtering, an original 3-D volume image can be enhanced by convolving with a 3-D sharpening operator of  $(M_1 \times M_2 \times 3)$  kernel. A three frame generic sharpening operator consists of  $(M_1 \times M_2 \times 3)$  coefficients of (-1), except the central element ( $m$ ) of the central frame which have the following value:

$$m = (-2) \times (-1) \times M_1 \times M_2 \times 3 - 1 \quad (5.15)$$

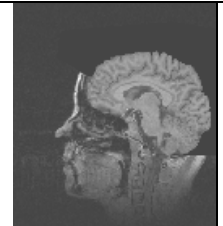
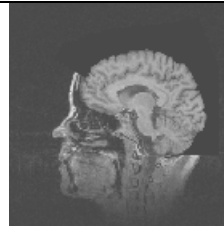
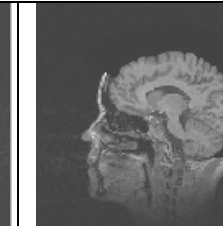
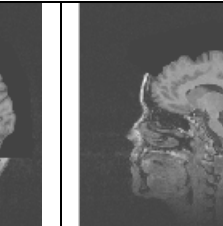
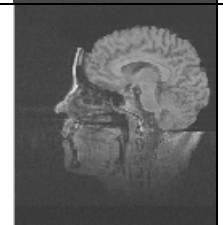

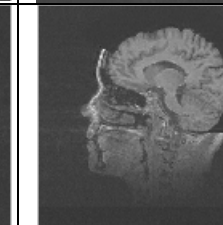
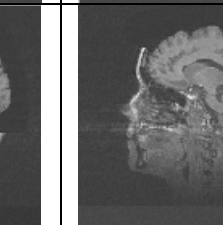


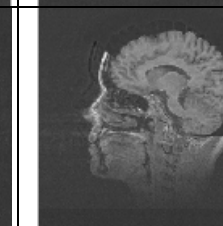
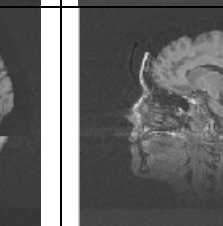
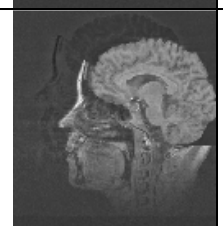
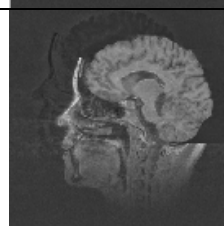
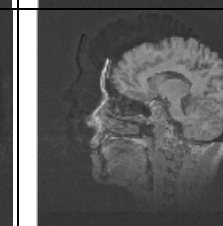
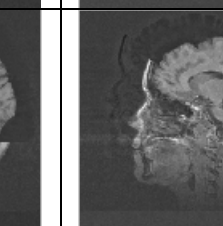
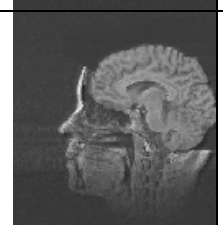
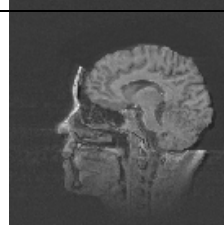
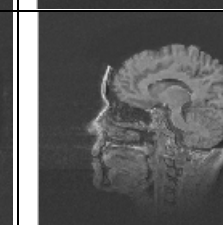
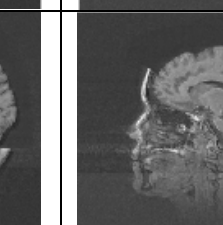
All elements to be divided by an operator factor ( $s = M_1 \times M_2 \times 3$ ) to ensure that the low spatial frequency is not amplified. The 3-D MRI volume data filtering results using architecture 21 and 22 respectively are shown in Table 5.19 and Table 5.20. The 3-D sharpen operator is a noise smoothing operator and Edge filtering to prevent the output value being different from the input in uniform regions of the MRI.

**Table 5.19: the filtering results for greyscale (64×64×8) MRI volume data of the generic 3-D FIR Sharpen operators using architecture 21**

3-D Sharpen Kernel $(M_1 \times M_2 \times 3)$				
3×3×3				
7×7×3				
15×15×3				
31×31×3				

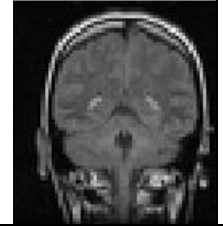
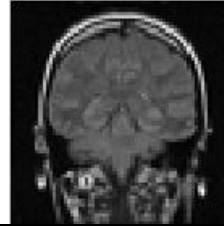
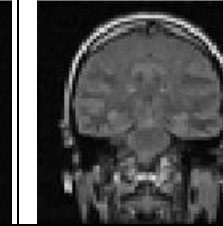
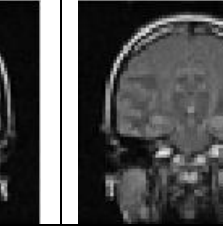
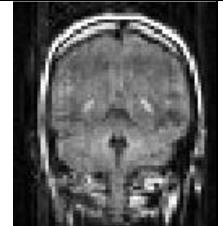
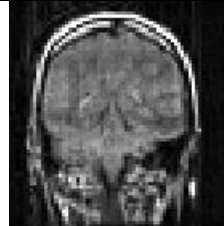
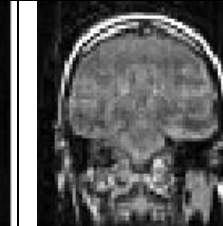
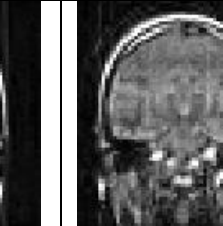

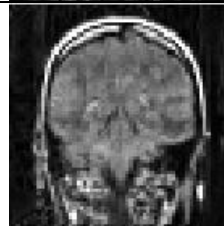
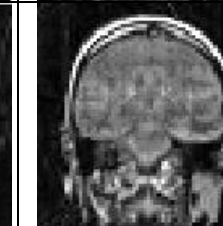
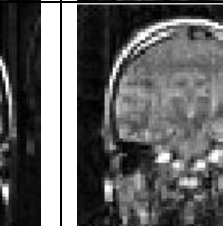

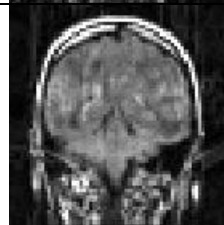
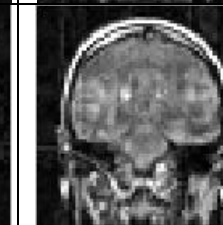

Architecture 22 was used to filter and visualize 3D ( $64 \times 64 \times 99$ ) k-Space MRI volume data input, producing a sagittal view of a human head as shown in Table 5.20. Where the original four slices of number 61 to 64, as shown on the top of the filtered results, are filtered and visualized out of a k-space MRI volume data stack of slices. Because of the way the data was collected there is some spatial aliasing in the reconstructed image for the Sharpen operator's kernel below  $31 \times 31 \times 3$  coefficients.

**Table 5.20: the sharpening results for k-space greyscale ( $64 \times 64 \times 8$ ) MRI volume data of the generic 3-D FIR Sharpen operators using architecture 22**

3-D Sharpen Kernel ( $M_1 \times M_2 \times 3$ )				
3×3×3				
7×7×3				
15×15×3				
31×31×3				

Architecture 23 can filter 3-D MRI volume data of double size than that filtered using architecture 21, due to the input 3-D MRI volume's decimation by 2. Then, the 3-D operator's kernel and the N-size FFT triplet has consequently half the value of that used in architecture 21 for the same MRI volume size. The filtered MRIs are slightly larger by  $(M_1-1 \times M_2-1 \times M_3-1)$  pixels with the filtered  $(N_1+M_1-1) \times (N_2+M_2-1) \times (N_3+M_3-1)$  image.

**Table 5.21: the filtering results for greyscale (64×64×8) MRI volume data of the generic 3-D FIR Sharpen operators using parallel architecture 23**

<p><b>3-D Sharpen Kernel</b> (<math>M_1 \times M_2 \times 3</math>)</p>				
<p>3×3×3</p>				
<p>7×7×3</p>				
<p>15×15×3</p>				

### 5.7 3-D Cross-Correlator Application: Parallel 3-D MRI Matched Filtering Algorithm

The 3-D FFT convolution can be used to evaluate the 3-D correlation of two 3-D images [19]. The detection of a targeted 3-D MRI volume from a 3-D MRI volume data library is presented for its diagnostic importance to access a particular part in the 3-D MRI volume. This similarity measure can be achieved by 3-D cross-correlation (3-D image match filter). To perform 3-D cross-correlation by using 3-D convolution, the target 3-D image needs to be reversed to counter-act the reversal that occurs during convolution. Thus, the 3-D MRI match filtering can be implemented using either architectures 21 or 23. To convert the target 3-D MRI slice into a 3-D match filtering operator, the targeted 3-D MRI volume flipped left-for right, top-for-bottom and frame-for-frame.

Therefore, architecture 21 and architecture 23 can be effectively used to be the implementation of the single 3-D correlator engine unit and the parallel 3-D MRI matched filtering algorithm respectively, to detect a targeted 3-D MRI volume

$h(m_1, m_2, m_3)$  within a 3-D MRI volume data library  $x(n_1, n_2, n_3)$ . Where, the 3-D FIR kernel is replaced by the target 3-D MRI volume input to be detected. Except, the target 3-D MRI volume has to be flipped, in three directions, left-for right, top-for-bottom and frame-for-frame to implement its reversed-time version  $h(-m_1, -m_2, -m_3)$ . For example, when  $x(n_1, n_2, n_3)$  and  $h(m_1, m_2, m_3)$ , are convolved to produce  $y_r(n_1, n_2, n_3)$  using architecture 23, the equation will be as:

$$y_r(n_1, n_2, n_3) = \sum_{r=0}^7 \left( \sum_{m_1=0}^{\frac{N_1}{2}-1} \sum_{m_2=0}^{\frac{N_2}{2}-1} \sum_{m_3=0}^{\frac{N_3}{2}-1} h(n_1 - m_1, n_2 - m_2, n_3 - m_3) x_r(m_1, m_2, m_3) \right) \quad (5.16)$$

where,  $r=0, 1 \dots 7$  are the parallel processing stages. In comparison, the correlation  $z(u)$  of  $x(n_1, n_2, n_3)$  and  $h(m_1, m_2, m_3)$  can be obtained by convolution of  $x(n_1, n_2, n_3)$  and  $h(-m_1, -m_2, -m_3)$  as following:

$$z_r(n_1, n_2, n_3) = \sum_{r=0}^7 \left( \sum_{m_1=0}^{\frac{N_1}{2}-1} \sum_{m_2=0}^{\frac{N_2}{2}-1} \sum_{m_3=0}^{\frac{N_3}{2}-1} h(n_1 - (-m_1), n_2 - (-m_2), n_3 - (-m_3)) x_r(m_1, m_2, m_3) \right) \quad (5.17)$$

That is, flipping left-for right in the  $N_1$  dimension, flipping top-for-bottom in the  $N_2$  dimension and flipping frame-for-frame in the  $N_3$  dimension are accomplished by reversing the sign of the time index. Accordingly, the parallel 3-D filtering described in (5.16) may be modified to perform parallel 3-D correlation filtering using (5.17).

## 5.8 Conclusion

A generalized parallel three/multi-dimensional image filtering algorithm was presented and implemented on Virtex-6 FPGA board. Eight generic architectures were developed, five as 3-D spatial convolution architectures, and three as 3-D FFT convolution architectures. The mathematical model of this parallel 3-D filtering algorithm was presented. Moreover, the mathematical model of the 3-D throughput was derived for the spatial and the frequency architectures. The performance indices of all the eight architectures were evaluated as a complete performance indices set of area, speed, dynamic power, throughput and the computation rate. Three successful applications were developed as four dimensions coloured MRI (fMRI) filtering processors, filter k-space MRI volume data and 3-D cross-correlator.

## Chapter 6. Conclusion and Future Work

### 6.1 Introduction

This research project was characterised by the adaptation of powerful parallel multi-dimensional data filtering algorithms and their highly efficient FPGA-based implementations, to be exploited for the linear filtering in 1-D, 2-D, 3-D and 4-D for computationally intensive applications of digital speech filtering, biomedical imaging filtration of greyscale MRI and multi-dimensional colour fMRI. The proposed parallel 1-D signal filtering algorithms were based on the well-known block filtering method. While, the proposed 2-D, 3-D and 4-D filtering algorithms were concurrently structured as the multi-dimension data segmentation by decimation and output reconstruction by interpolation.

Twenty-three generic architectures were designed to efficiently realise the parallel multi-dimensional data convolution/correlation algorithms in the time and frequency domains, depending on the multi-dimensional filter's kernel size. These parallel multi-dimensional architectures were structured as 1-D, 2-D, 3-D and 4-D spatial and FFT convolver engines. The FPGA implementations aim at achieving highly efficient performance indices as a complete set of area, speed, dynamic power, throughput and the computation rate. The implementations' performance indices were obtained using the available logic assets of the targeted Virex-6 XC6VLX240T development FPGA board. Furthermore, their efficiency is demonstrated by the performance indices results comparison that highlights the superiority of the hardware versions of the proposed parallel multi-dimensional data filtering algorithms.

### 6.2 Original Contributions

The overall contributions that have been achieved and presented within this thesis are summarised by a short list for each chapter.

#### *6.2.1 Parallel 1-D Signal Filtering Architectures*

In chapter 3, parallel one dimension filtering algorithm was proposed and mathematically modelled using the overlap-add block filtering method. This parallel algorithm was implemented on the FPGA, as parallel temporal and spectral architectures, which achieved a set of values for the high performance indices. Five parallel architectures were developed based on three 1-D temporal convolvers and one 1-D FFT convolver engines, as follows:

- The mathematical model of the input, processing and the output stages of the parallel 1-D linear filtering algorithm is presented.
- A generalized parallel hardware versions for the 1-D linear filtering algorithm are developed for the multi-MAC FIR convolution and FFT convolution, to cover the entire range of linear FIR filter length of 2, 3, 4, 5, 7, 8, 15, 16, 31, 32, 63, 64, 127, 255, 511, 1023, 2047, ... coefficients.
- Five FPGA-based architectures on Virtex-6 ML605 board are developed
- Three new temporal convolver engines are developed and implemented using single MAC, dual MAC and quad MAC units respectively
- Single 1-D FFT filtering algorithm and parallel 1-D FFT filtering algorithm are implemented using the linear overlap-add block filtering method.
- A successful application of cross-correlation using FFT convolution is realized.
- The complete set of area, speed, power, throughput and computation rate in this chapter are compared and discussed as the performance indices for the five architectures.

### ***6.2.2 Parallel 2-D Grayscale/Colour Image Filtering Architectures***

In chapter 4, parallel two/three dimensions filtering algorithm was proposed and mathematically modelled using image segmentation by decimation and output reconstruction by interpolation. This parallel algorithm was implemented on Virtex-6 FPGA as parallel spatiotemporal and spectral architectures which achieved a highly performance indices results. These parallel architectures were ten, based on three 2-D spatiotemporal convolvers and one 2-D FFT convolver engines, as follows:

- A generalized parallel 2-D image filtering algorithm is presented with its mathematical model.
- To cover the entire range of linear 2-D image filtering, a generic parallel hardware versions of the 2-D linear image filtering algorithm are developed for the multi-MAC FIR and FFT convolution.
- Ten generic architectures of 2-D image processors on Virtex-6 ML605 board are implemented. Their complete performance indices set are evaluated.
- Architectures (6, 7 and 8), architectures (9 and 10) and Architectures (11 and 12) are implemented using single, dual and quad MAC convolver engines respectively.
- 3-D Colour image processor is devised to act as an open development 3-D colour filtering engines using Architectures 6, 7, 8, 9, 10, 11 and 12.

- Architectures (13 and 15) are implemented to capture the single 2-D FFT convolver unit and the parallel 2-D FFT convolution algorithm respectively.
- Architecture 14 is particularly developed to deal with real-time k-space MRI data.
- Cross-correlator parallel engine is successfully developed as a parallel 2-D matched filter algorithm to map a parallel 2-D cross-correlation algorithm to locate any MRI slice within a MRI data stack library.

### ***6.2.3 Parallel 3-D Grayscale/Colour Image Filtering Architectures***

In chapter 5, parallel 2-D/3-D convolution algorithm was proposed and mathematically modelled using volumetric data image segmentation by decimation and output reconstruction by interpolation. Then, FPGA implemented, as parallel spatial and spectral architectures, were achieved with a set of highly performance indices results. These parallel architectures were eight, based on three 3-D spatial convolvers and one 3-D FFT convolver engines, as follows:

- A generalized parallel three dimensional grayscale/colour image filtering algorithm is proposed and mathematically modelled.
- The parallel multi-dimensional image filtering algorithm is successfully implemented on eight generic architectures.
- The performance indices of the eight architectures are evaluated as a complete set of area, speed, dynamic power, throughput and, additionally, the computation rate for the spatial architectures.
- The five implementations of the parallel 3-D spatial convolution algorithm are devised as “plug and develop” architectures to filter a 3-D volume data.
- Fourteen 3-D MRI filtering operators, edge and noise smoothing, are plugged and developed to improve the suitability for biomedical imaging.
- Three successful applications are developed as 4-D coloured MRI (fMRI) filtering processors, filter k-space MRI volume data and 3-D cross-correlator.

### **6.3 Overall Conclusions**

The overall conclusions that summarize the achievements in proposing the parallel multi-dimensional data filtering algorithms and the implementation of the twenty-three FPGA-based architectures were highly demonstrated by the efficient performance indices.



### ***6.3.1 Performance Indices of the Parallel Multi-Dimensional Architectures***

In chapter 2, the main aspects of parallelism in the parallel multi-dimensional data filtering algorithms and their realization using the inherent parallelism of the FPGA that improves the performance efficiency indices in term of area, speed, power and throughput. As summarised below:

1. The implementation suitability of the inherently parallel FPGA to map into hardware architectures for the parallel mutli-dimensions filtering algorithms was presented
2. Parallelism either existed intrinsically in the parallel multi-dimensional filtering algorithms or could be introduced by organizing the computation to allow a parallel implementation. Three parallelisms were observed in these filtering algorithms: temporal, spatial and logical parallelisms.
3. Parallel streaming processing can be efficiently realized in the FPGA.
4. The FPGA-based implementation was characterized by the flexibility to adjust the wordlength size to achieve the necessary arithmetic resolution, which can be changed at different parts of the inherent parallelism hardware.
5. FPGAs enable the configuration of data paths into arbitrary wordlength sizes, allowing a trade-off between precision and parallelism. An additional benefit of minimizing precision comes from shorter propagation delays through narrower arithmetic units.
6. The overall FPGA implementation process was depicted as a development flow of seven key steps.
7. Parallel Field Programming using Xilinx System Generator was discussed as a dataflow graphical programming tool that facilitates FPGA hardware design by providing access to underlying FPGA resources.
8. The performance indices of the parallel multi-dimensional data filtering architectures were considered as a complete set of area, speed, power, throughput and computation rate values using XSG to target a Virtex-6 ML605 FPGA board.
9. The area occupation, dynamic power consumption and speed were obtained using Xilinx Timing Analyzer. The throughput and computation rate were calculated according to a 1-D, 2-D and 3-D mathematical model.

### ***6.3.2 Parallel 1-D Signal Filtering Architectures***

In chapter 3, the generalized parallel 1-D data convolution/correlation algorithms were adapted, mathematically modelled, implemented, analysed and discussed. This can be summarised as follows:

1. The mathematical model of the input, processing and the output stages of the parallel 1-D filtering algorithm was mathematically presented.
2. A generalized parallel hardware versions of the parallel 1-D convolution-filtering algorithm was developed for the temporal and FFT convolution, to

cover the whole range of linear FIR filter lengths of 3, 7, 15, 31, 61, 127, 255, 511, 1023, 2047, ... coefficients. The filtering methods were based on the overlap-add block filtering.

3. Five FPGA-based architectures were realized on Virtex-6 ML605 board, three for the temporal convolution and two for the FFT convolution.
4. A successful applications was presented to realize cross-correlation as a parallel 1-D matched filter algorithm for real-time speech signature detection.
5. The performance indices for the six architectures were considered as a complete set of area, speed, power and throughput.
6. The three temporal architectures had high throughput, stable maximum clock frequency and low dynamic power consumption with a distinctive computation rate.
7. The highest throughput, computation rate and the lowest dynamic power consumption was achieved by architecture 3 and 1 respectively.
8. The three implementations of the parallel 1-D MAC convolution algorithm were developed as “plug and filter” architectures.
9. Architecture 5’s throughput outperformed, in average, by four-fold than that of architecture 4, due to the embedded parallelism.

### ***6.3.3 Parallel 2-D Grayscale/Colour Image Filtering Architectures***

In chapter 4, the generalized parallel 2-D image convolution/correlation algorithms were adapted, mathematically modelled, implemented, analysed and discussed, and applied to the biomedical imaging of grayscale/colour MRI (fMRI). This can be summarised as follows:

1. To cover the extensive range of linear 2-D Image filtering and the 2-D image, in particular large MRI slice size, generalized parallel hardware versions of the 2-D convolution algorithm was devised, analysed and implemented on Virtex-6 development board for the spatiotemporal and FFT convolution.
2. For the linear FIR filters’ kernel shorter than about  $10 \times 10$ , the spatiotemporal convolution was developed for  $2 \times 2$ ,  $3 \times 3$ ,  $5 \times 5$ ,  $2 \times 4$ ,  $4 \times 4$ ,  $2 \times 8$  and  $8 \times 8$  kernels. Beyond these kernels, FFT convolution was developed for the longer filter kernel sizes of  $15 \times 15$ ,  $31 \times 31$ ,  $63 \times 63$ ,  $127 \times 127$  and  $255 \times 255$ . The filtered MRI slice was of different sizes from  $64 \times 64$ ,  $128 \times 128$  and  $256 \times 256$  up to  $512 \times 512$  pixels. The MRI data input was either digital image or real-time k-space data.
3. The parallel 2-D filtering algorithm was devised by decimating the 2-D input data into independent sub-data that can be simultaneously processed in parallel without the need for processing stage data communication, the final filtered result was reconstructed by 2-D interpolation and right shift. The mathematical model was presented.
4. Ten FPGA-based architectures were developed, seven for spatiotemporal convolution and three for the FFT convolution.

5. Architectures (6, 7 and 8), architectures (9 and 10) and architectures (11 and 12) were implemented to realize the spatiotemporal convolution, while, architectures (13 and 15) were realized as the FFT convolution.
6. The performance indices for the ten architectures were evaluated as a complete set.
7. The superiority of the developed architectures were indicated by the minimized utilized area, high throughput, stable maximum clock frequency and low dynamic power consumption. These FPGA-architectures were strongly exploited as a “plug and develop” reconfigurable image processor.
8. Twelve improved 2-D FIR filters were plugged and developed to filter for edge detection and noise smoothing of the biomedical imaging of grayscale and colour MRI data.
9. Three successful applications on medical image filtering and detection were presented to illustrate the superiority and high performance of the parallel 2-D convolution algorithm. These three applications were 3-D colour MRI (fMRI) filtering, real-time k-space MRI data visualization and detection of a MRI slice within a MRI data stack library.

#### ***6.3.4 Parallel 3-D Grayscale/Colour Image Filtering Architectures***

In chapter 5, a generalized parallel three/multi-dimensional image filtering algorithm was proposed and implemented on Virtex-6 development board. This new parallel algorithm eliminates the overhead associated with the overlapping segments in the block filtering method and the boundary conditions in the parallel filtering implementation. Moreover, the normal large size restriction of multi-dimensional input data was overcome by decimation into smaller, manageable and independent multi-dimensions sub-images. This can be summarised as follows:

1. The proposed parallel algorithm’s mathematical model was derived.
2. This parallel algorithm was successfully implemented on eight hardware architectures to cover the wide-ranging 3-D FIR kernels, five as 3-D spatial convolution architectures, and three as 3-D FFT convolution architectures. These 3-D architectures were found to be highly parallel and superior in utilising the inherent parallelism advantages of the FPGA.
3. The performance indices of all the eight architectures were evaluated as a complete set of area, speed, dynamic power, throughput and the computation rate. Noteworthy improvements had been achieved in decreasing power consumption and increasing throughput of more than 4-fold and 16-fold respectively.
4. The massive parallelism filtering implementations were demonstrated by minimum logic area occupation, low dynamic power consumption and high throughput at maximum clock frequency.

5. The five implementations of the parallel 3-D spatial convolution algorithm were devised as “plug and develop” architectures to filter a 3-D grayscale/colour volumetric data.
6. Fourteen 3-D MRI filtering operators were plugged and developed to be improved for their suitability to biomedical imaging applications.
7. Three successful applications were presented, one for the spatial architectures as four dimension coloured MRI (fMRI) filtering processors, and two for the 3-D FFT architectures to filter k-space MRI volume data and 3-D cross-correlator.

#### **6.4 Recommendation for Future Work**

Based on the experience and knowledge acquired during this research project, the recommendation for future research projects can be suggested as a complete automated filtering system. The VLSI implementation of all the twenty-three architectures as a complete parallel multi-dimensional data filtering set is highly recommended, based on input segmentation, parallel independent multi-dimensional filtering and output reconstruction. This multi-dimensional filtering automation system can be parameterized in the following aspects: firstly, the dimension of the input data; is it 1-D, 2-D, 3-D or 4-D? Secondly, the application type: is it biomedical imaging, seismic data, marketing data stream ...etc? Thirdly, filtering type method: Is it spatial or spectral filtering? based on the linear filter length. Fourthly, performance indices optimized to be high throughput, low power, high speed, low area occupation.

IP hard cores of performance-aware fast switching circuitries may control the data routing and the I/O interface of this complete real time firmware filtering system. The performance-aware optimization is application dependent to be concerned with low-power consumption, high filtering rate and/or high throughput. This research extension may lead to new and novel VLSI generic architectures for on-the-fly parallel multidimensional data processing, aiming at higher data routing rate and optimum memory layout in order to solve the bottleneck of multi-dimensional data processing with normal large input size, memory limitation and multi-dimensional convolver/correlator engines.

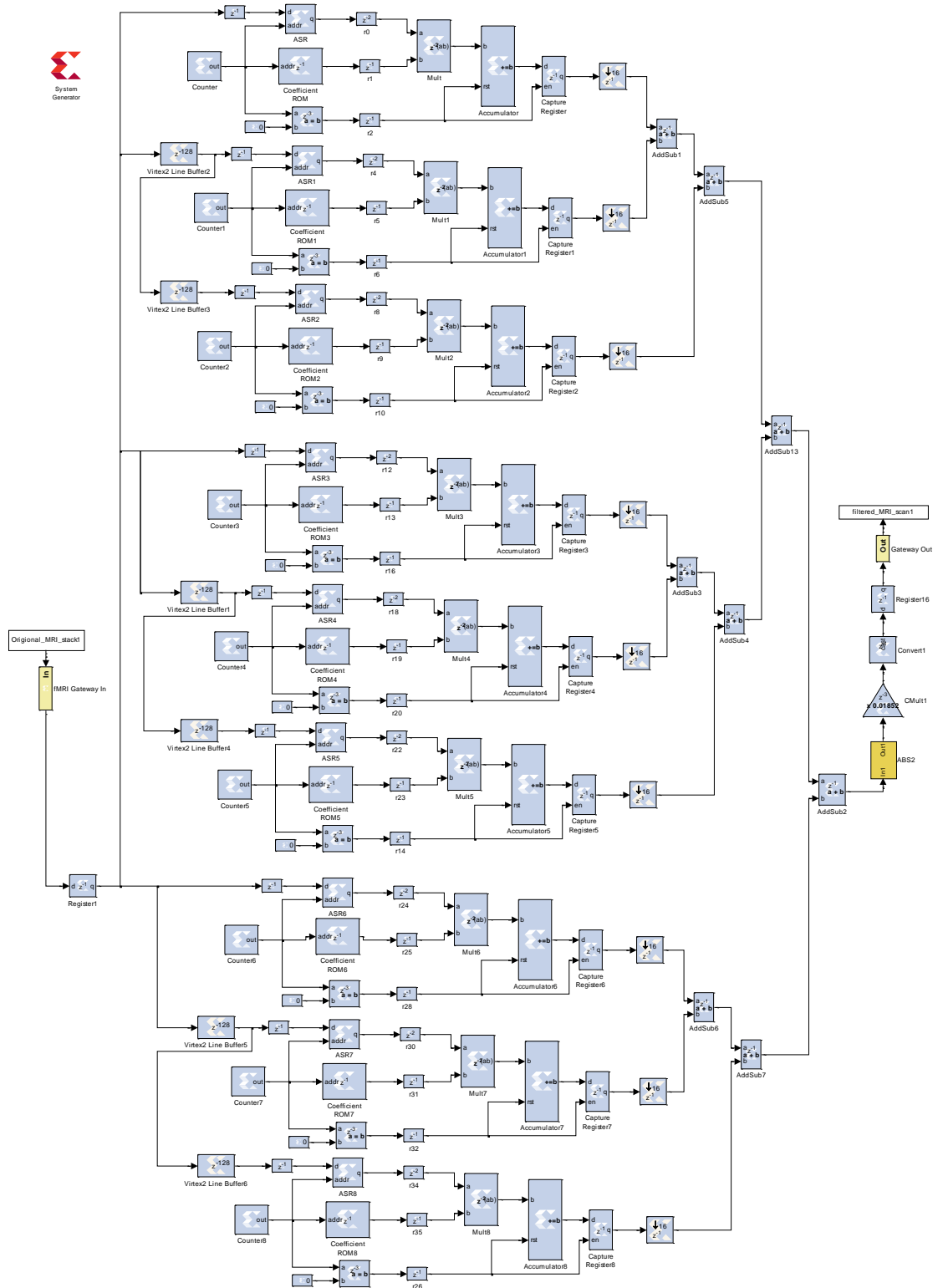
One of the recent techniques in low power VLSI is to tolerate the computation's accuracy for the power consumption that can be effectively utilized to trade off power and quality for error-resilient DSP systems [131]. Low power is an imperative requirement for portable multimedia devices employing various signal processing algorithms and architectures. In most multimedia applications, human beings can gather

useful information from slightly erroneous outputs. Therefore, we do not need to produce exactly correct numerical outputs.

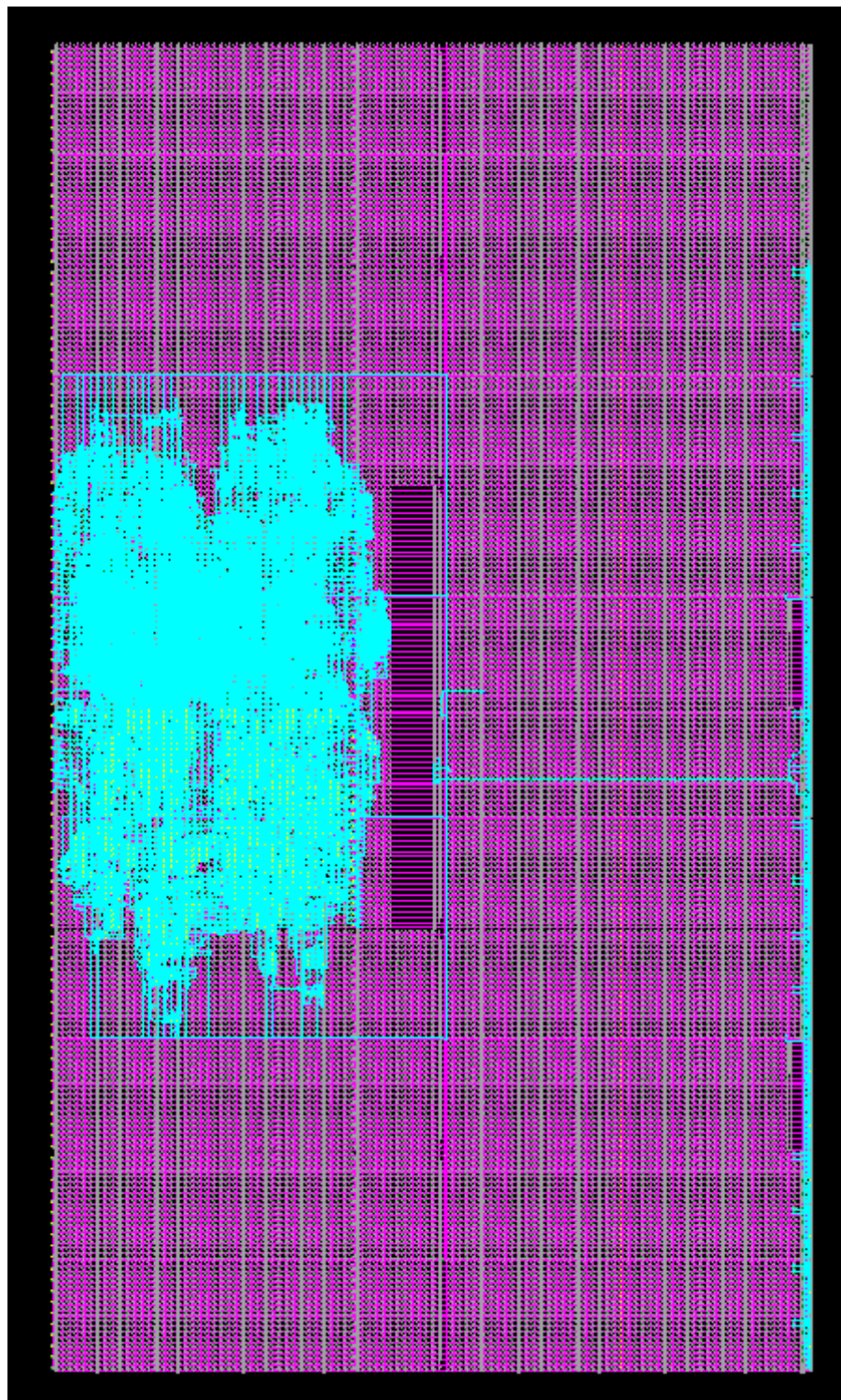
Finally, the investigation is always in pursuit of new and novel reconfigurable parallel hardware that can be field developed by exploiting parallel programming tools in accelerating applications in major areas of video processing using reconfigurable VLSI architectures. Video processing is highly amenable to parallel processing, computationally intensive and often has accompanying real-time or super-real-time requirements, which broadly encompasses compression, filtering enhancement, analysis, and synthesis of digital video. Consequently, the research and development of massively parallel architectures and programming technology in the construction and development of parallel multi-dimensional data processing components and applications, is highly recommended. For example, surveillance and monitoring systems need to robustly analyze video from multiple cameras in real time to automatically detect and signal unusual events. Beyond today's known applications, the continued growth of functionality and speed of video processing systems will likely further enable novel applications.

# APPENDICES

## Appendix A. Architecture 17's first filtering stage out of eight

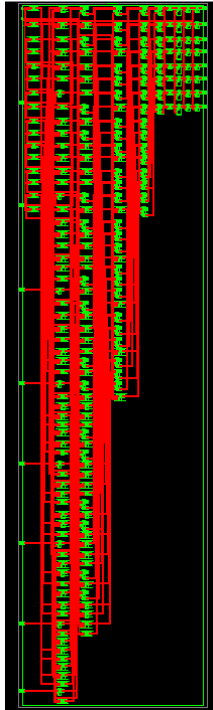


**Appendix B. Logic area occupation of architecture 17 mapped and shown graphically using Xilinx FPGA Editor Tool**

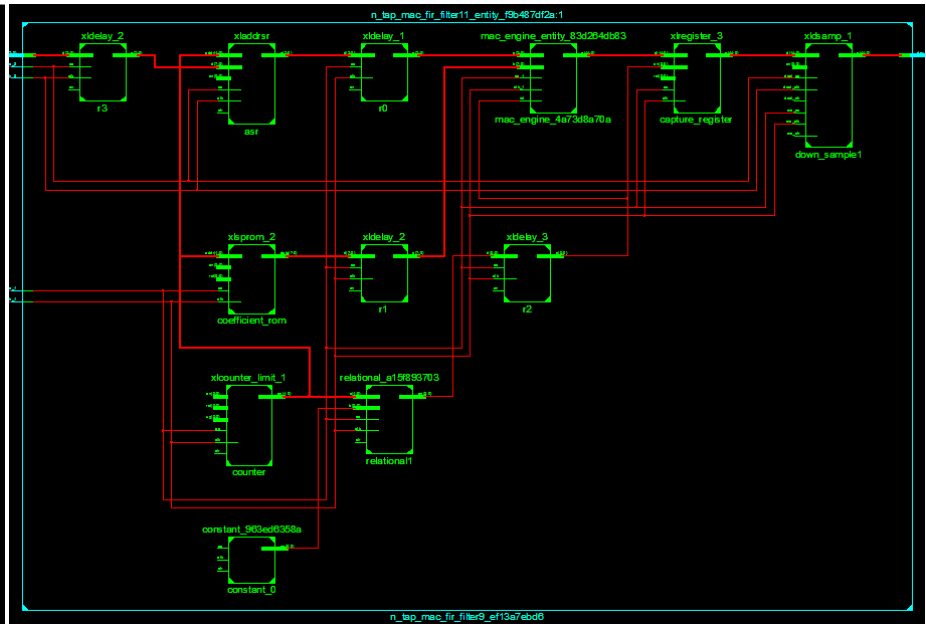


## Appendix C. Architecture 17's RTL schematic diagram

(1)



(2)



- 1) Shows the entire RTL schematic diagram
- 2) Shows partially enlarged RTL schematic diagram



# List of Publications

## Journal Publications

- [1] S. Hasan, S. Boussakta, and A. Yakovlev, "FPGA-Based Architecture for a Generalized Parallel 2-D MRI Filtering Algorithm," *American Journal of Engineering and Applied Sciences*, vol. 4, pp. 566-575, 2012.

## Conference publications

- [1] S. Hasan, S. Boussakta, and A. Yakovlev, "Improved parameterized efficient FPGA implementations of parallel 1-D filtering algorithms using Xilinx System Generator," in *Signal Processing and Information Technology (ISSPIT), 2010 IEEE International Symposium on*, 2010, pp. 382-387.
- [2] S. Hasan, S. Boussakta, and A. Yakovlev, "Parameterized FPGA-based architecture for parallel 1-D filtering algorithms," in *Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop on*, 2011, pp. 171-174.
- [3] S. Hasan, A. Yakovlev, and S. Boussakta, "Performance efficient FPGA implementation of parallel 2-D MRI image filtering algorithms using Xilinx system generator," in *Communication Systems Networks and Digital Signal Processing (CSNDSP), 2010 7th International Symposium on*, 2010, pp. 765-769.

## REFERENCES

- [1] J. F. Michael and W. R. Kevin, "Parallel architectures," *ACM Comput. Surv.*, vol. 28, pp. 67-70, 1996.
- [2] Z. Wang, M. Gao, X. Fu, and C. Jiang, "Design of Real-time Convolution Processor and its Application in Radar Echo Signal Simulator," in *International Conference on Computer Science and Information Technology, ICCSIT '08*, 2008, pp. 162-166.
- [3] I. S. Uzun, A. Amira, and F. Bensaali, "A reconfigurable coprocessor for high-resolution image filtering in real time," in *10th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2003*, pp. 192-195 Vol.1.
- [4] M. Sankaradas, V. Jakkula, S. Cadambi, S. Chakradhar, I. Durdanovic, E. Cosatto, and H. P. Graf, "A Massively Parallel Coprocessor for Convolutional Neural Networks," in *20th IEEE International Conference on Application-specific Systems, Architectures and Processors, ASAP 2009*, pp. 53-60.
- [5] L. M. Russo, E. C. Pedrino, E. Kato, and V. O. Roda, "Image convolution processing: A GPU versus FPGA comparison," in *2012 VIII Southern Conference on Programmable Logic (SPL)* pp. 1-6.
- [6] N. K. Ratha, A. K. Jain, and D. T. Rover, "Convolution on Splash 2," in *1995 IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 204-213.
- [7] H. Pei-Yung, W. Hwa, C. Ying-Pei, and C. Sao-Jie, "Real-time implementation of noise-immune gradient-based edge detection," in *International Symposium on Signals, Circuits and Systems, ISSCS 2005* pp. 633-636 Vol. 2.
- [8] A. Nieto, V. M. Brea, and D. L. Vilarino, "FPGA-accelerated retinal vessel-tree extraction," in *International Conference on Field Programmable Logic and Applications, FPL 2009* pp. 485-488.
- [9] J. Y. Mori, C. Sanchez-Ferreira, Mu, x00F, D. M. oz, C. H. Llanos, and P. Berger, "An unified approach for convolution-based image filtering on reconfigurable systems," in *2011 VII Southern Conference on Programmable Logic (SPL)* pp. 63-68.
- [10] P. Longa, A. Miri, and M. Bolic, "A Flexible Design of Filterbank Architectures for Discrete Wavelet Transforms," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2007*, pp. III-1441-III-1444.
- [11] P. Y. Hsiao, C. H. Chen, H. Wen, and S. J. Chen, "Real-time realisation of noise-immune gradient-based edge detector," *Computers and Digital Techniques, IEE Proceedings -*, vol. 153, pp. 261-269, 2006.
- [12] N. Farrugia, F. Mamalet, S. Roux, Y. Fan, and M. Paindavoine, "Fast and Robust Face Detection on a Parallel Optimized Architecture Implemented on FPGA," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, pp. 597-602, 2009.
- [13] N. Farrugia, F. Mamalet, S. Roux, Y. Fan, and M. Paindavoine, "A Parallel Face Detection System Implemented on FPGA," in *IEEE International Symposium on Circuits and Systems, ISCAS 2007*, pp. 3704-3707.
- [14] A. S. Dawood, S. J. Visser, and J. A. Williams, "Reconfigurable FPGAS for real time image processing in space," in *14th International Conference on Digital Signal Processing, DSP 2002*, pp. 845-848 vol.2.
- [15] S. Boussakta, "A novel method for parallel image processing applications," *Journal of Systems Architecture*, vol. 45, pp. 825-839, 1999.
- [16] Roger Woods, Jhon McAllister, Y. Yi, and G. Lightbody, *FPGA-based implementation of Signal Processing Systems: Join Wiley & Sons Ltd*, 2008.
- [17] D. G. Bailey, *Design for Embedded Image Processing on FPGA: Join Wiley & sons Pte Ltd*, 2011.

- [18] J. L. Semmelow, *Biosignal and Biomedical Image Processing*: Marcel Dekker, Inc., 2004.
- [19] N. Nikolaidis and I. Pitas, *3-D Image Processing Algorithms*: John Wiley & Sons, Inc., 2001.
- [20] R. Miller and L. Boxer, *Algorithms Sequential and Parallel: A Unified Approach*, 2nd ed.: Charles River Media Computer Engineering, 2005.
- [21] S. Hasan, A. Yakovlev, and S. Boussakta, "Performance efficient FPGA implementation of parallel 2-D MRI image filtering algorithms using Xilinx system generator," in *7th International Symposium on Communication Systems Networks and Digital Signal Processing (CSNDSP 2010)* pp. 765-769.
- [22] R. Zoss, A. Habegger, V. Bandi, J. Goette, and M. Jacomet, "Comparing Signal Processing Hardware-Synthesis Methods Based on the Matlab Tool-Chain," in *2011 Sixth IEEE International Symposium on Electronic Design, Test and Application (DELTA)*, pp. 281-286.
- [23] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, Design Tools, and Application Domains of FPGAs," *Industrial Electronics, IEEE Transactions on*, vol. 54, pp. 1810-1823, 2007.
- [24] H. James, M. Brent, S. Nabeel, and D. S. Jeffrey, "System Level Tools for DSP in FPGAs," in *the 11th International Conference on Field-Programmable Logic and Applications*, 2001.
- [25] M. Aziz, S. Boussakta, and D. C. McLernon, "Parallelisation of the 1-D block filter algorithm to run on multiple DSPs," in *9th International Conference on Electronics, Circuits and Systems*, 2002, pp. 943-946
- [26] S. Hasan, S. Boussakta, and A. Yakovlev, "Parameterized FPGA-based architecture for parallel 1-D filtering algorithms," in *7th International Workshop on Systems, Signal Processing and their Applications (WOSSPA 2011)*, pp. 171-174.
- [27] S. Hasan, S. Boussakta, and A. Yakovlev, "Improved parameterized efficient FPGA implementations of parallel 1-D filtering algorithms using Xilinx System Generator," in *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2010)*, pp. 382-387.
- [28] P. Sungho, Y. C. P. Cho, K. M. Irick, and V. Narayanan, "A reconfigurable platform for the design and verification of domain-specific accelerators," in *2012 17th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 108-113.
- [29] S. M. A. Raj, G. Sreelatha, and M. H. Supriya, "Gesture recognition using field programmable gate arrays," in *2012 International Conference on Devices, Circuits and Systems (ICDCS)* pp. 72-75.
- [30] V. R. Pamula, W. Strauss, and J. Bernhard, "Model Based Development of the Digital Part of a RFID Transponder with Xilinx System Generator for a FPGA Platform," in *2012 Fourth International EURASIP Workshop on RFID Technology (EURASIP RFID)*, pp. 124-127.
- [31] J. Hwang and J. Ballagh, "Building custom FIR filters using system generator," in *12th International Conference on Field-Programmable Logic and Applications, FPL 2002*, pp. 1101-1104.
- [32] M. Aziz, S. Boussakta, and D. C. McLernon, "High performance 2D parallel block-filtering system for real-time imaging applications using the Sharc ADSP21060," *Real-Time Imaging*, vol. 9, pp. 151-161, 2003.
- [33] I. S. Uzun, A. Amira, and A. Bouridane, "FPGA implementations of fast Fourier transforms for real-time signal and image processing," *Vision, Image and Signal Processing, IEE Proceedings -*, vol. 152, pp. 283-296, 2005.

- [34] W. Atabany and P. Degenaar, "Parallelism to reduce power consumption on FPGA spatiotemporal image processing," in *IEEE International Symposium on Circuits and Systems, (ISCAS 2008)*, pp. 1476-1479.
- [35] S. Hasan, S. Boussakta, and A. Yakovlev, "FPGA-Based Architecture for a Generalized Parallel 2-D MRI Filtering Algorithm," *American Journal of Engineering and Applied Sciences*, vol. 4, pp. 566-575, 2012.
- [36] S. Athar, M. A. Siddiqi, and S. Masud, "Teaching and research in FPGA based Digital Signal Processing using Xilinx System Generator," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2012)* pp. 2765-2768.
- [37] T. Mahalakshmi and R. Muthaiah, "Vlsi Implementation of Edge Detection for Images," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, pp. 5474-5479, 2012.
- [38] S. Allin Christe, Vignesh, M. and Kandaswamy, A., "An efficient FPGA implementation of MRI image filtering and tumour characterization using Xilinx System Generator," *International Journal of VLSI design & Communication Systems (VLSICS)*, vol. Vol.2, pp. 96-109, 2011.
- [39] M. Aziz and S. Boussakta, "Efficient 3-D parallel fir filtering algorithm," in *15th European Signal Processing Conference, EUSIPCO 2007*, pp. 1064-1067.
- [40] M. Aziz, S. Boussakta, and D. C. McLernon, "Three-dimensional digital filtering algorithm for parallel DSP implementation," in *International Conference on Image Processing, ICIP 2003* pp. II-579-82 vol.3.
- [41] M. Aziz, S. Boussakta, and D. C. McLernon, "A high performance 3-D parallel filtering algorithm using the vector radix fast Hartley transform," in *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT'2003*, pp. 715-719.
- [42] M. Aziz, D. C. McLernon, and S. Boussakta, "The implementation of a new 3-D parallel filtering algorithm on the SHARC ADSP21060 platform," in *International Conference on Visual Information Engineering, VIE 2003*, pp. 270-273.
- [43] T. B. Deng, "Decomposition-based design of linear phase 3-D digital filters," *Signal Processing*, vol. 54, pp. 119-128, 1996.
- [44] D. Lin, H. Xiaohuang, N. Quang, J. Blackburn, C. Rodrigues, T. Huang, M. N. Do, S. J. Patel, and W. M. W. Hwu, "The parallelization of video processing," *Signal Processing Magazine, IEEE*, vol. 26, pp. 103-112, 2009.
- [45] A. Naoulou, J. L. Boizard, J. Y. Fourniols, and M. Devy, "A 3D real-time vision system based on passive stereovision algorithms: Application to laparoscopic surgical manipulations," in *2nd Information and Communication Technologies, ICTTA '06*, pp. 1068-1073.
- [46] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Third Edition ed., 2007.
- [47] C. Maxfield, *FPGAs World Class Designs*: Newnespress/ Elsevier, 2009.
- [48] C. Maxfield, *The Design Warrior's Guide to FPGAs*: newnespress/ Elsevier, 2004.
- [49] I. Grout, *Digital Systems Design with FPGAs and CPLDs*: newnespress/ Elsevier, 2008.
- [50] G. Goavec-Merou, Y. Yakoubi, R. Couturier, M. Lenczner, J. M. Friedt, and F. Wang, "FPGA Implementation of Diffusive Realization for a Distributed Control Operator," in *First Workshop on Hardware and Software Implementation and Control of Distributed MEMS (DMEMS 2010)*, pp. 50-55.
- [51] N. Viscogliosi, J. Riendeau, P. Bérard, M. A. Tétrault, R. Lefebvre, R. Lecomte, and R. Fontaine, "Real time implementation of a wiener filter based crystal

- identification algorithm," *IEEE Transactions on Nuclear Science*, vol. 55, pp. 925-929, 2008.
- [52] T. F. Neves, M. F. Caetano, and J. L. Bordim, "An Energy-Optimum and Communication-Time Efficient Protocol for Allocation, Scheduling and Routing in Wireless Networks," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, pp. 848-854.
- [53] W. G. Gardner, "Efficient convolution without input-output delay," *AES: Journal of the Audio Engineering Society*, vol. 43, pp. 127-136, 1995.
- [54] R. Mahesh and A. P. Vinod, "New Reconfigurable Architectures for Implementing FIR Filters With Low Complexity," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, pp. 275-288, 2010.
- [55] L. R. Johnson and A. K. Jain, "An Efficient Two-Dimensional FFT Algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-3, pp. 698-701, 1981.
- [56] S. K. Madishetty, A. Madanayake, R. J. Cintra, V. S. Dimitrov, and D. H. Mugler, "VLSI Architectures for the 4-Tap and 6-Tap 2-D Daubechies Wavelet Filters Using Algebraic Integers," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. PP, pp. 1-1, 2012.
- [57] S. Nedunuri, J. Y. Cheung, and P. Nedunuri, "Design of Low Memory Usage Discrete Wavelet Transform on FPGA Using Novel Diagonal Scan," in *International Symposium on Parallel Computing in Electrical Engineering, PAR ELEC 2006* pp. 192-197.
- [58] L. Xuguang, Z. Nanning, and L. Yuehu, "Low-power and high-speed VLSI architecture for lifting-based forward and inverse wavelet transform," *Consumer Electronics, IEEE Transactions on*, vol. 51, pp. 379-385, 2005.
- [59] M. Ye, W. H. Zhou, and W. K. Gu, "FPGA based real-time image filtering and edge detection," *Chinese Journal of Sensors and Actuators*, vol. 20, pp. 623-627, 2007.
- [60] J. A. Webb, "Steps toward architecture-independent image processing," *Computer*, vol. 25, pp. 21-31, 1992.
- [61] K. S. Reddy and B. M. Bhaskara, "Noval approach image processing algorithms on hardware implementation for surveillance systems," *International Journal of Science & Technology*, vol. 1, 2011.
- [62] S. Haoyu, H. Fang, M. Kodialam, and T. V. Lakshman, "IPv6 Lookups using Distributed and Load Balanced Bloom Filters for 100Gbps Core Router Line Cards," in *IEEE INFOCOM 2009*, pp. 2518-2526.
- [63] E. A. Rivera-Juarico, J. M. Ramirez-Cortes, V. Alarcon-Aquino, and J. Escamilla-Ambrosio, "Design and implementation of the discrete wavelet transform on an FPGA platform to process data sets of up to three dimensions," in *2012 22nd International Conference on Electrical Communications and Computers (CONIELECOMP)* pp. 333-338.
- [64] B. Das and S. Banerjee, "A memory efficient 3-D DWT architecture," in *16th International Conference on VLSI Design*, 2003, pp. 208-213.
- [65] S. Ferrari and N. A. Borghese, "Portable modular system for automatic acquisition of 3D objects," *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, vol. 3, pp. 1823-1827, 1999.
- [66] H. Scherl, M. Kowarschik, H. G. Hofmann, B. Keck, and J. Hornegger, "Evaluation of state-of-the-art hardware architectures for fast cone-beam CT reconstruction," *Parallel Computing*, vol. 38, pp. 111-124, 2012.

- [67] P. Pahalawatta and K. Stec, "A subjective comparison of depth image based rendering and frame compatible stereo for low bit rate 3D video coding," in *2012 Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1-7.
- [68] M. C. Herbordt, T. VanCourt, Y. Gu, B. Sukhwani, A. Conti, J. Model, and D. DiSabello, "Achieving High Performance with FPGA-Based Computing," *Computer*, vol. 40, pp. 50-57, 2007.
- [69] P. Naik, M. Wedel, L. Bacon, A. Bodapati, E. Bradlow, W. Kamakura, J. Kreulen, P. Lenk, D. M. Madigan, and A. Montgomery, "Challenges and opportunities in high-dimensional choice data analyses," *Marketing Letters*, vol. 19, pp. 201-213, 2008.
- [70] S. Masuno, T. Maruyama, Y. Yamaguchi, and A. Konagaya, "Multidimensional dynamic programming for homology search," in *2005 International Conference on Field Programmable Logic and Applications* pp. 173-178.
- [71] M. Sheliga, N. L. Passos, and S. Edwin Hsing-Mean, "Fully parallel hardware/software codesign for multi-dimensional DSP applications," in *Fourth International Workshop on Hardware/Software Co-Design, Codes/CASHE '96* pp. 18-25.
- [72] X. Li and G. Qian, "Block size considerations for multidimensional convolution and correlation," *Signal Processing, IEEE Transactions on*, vol. 40, pp. 1271-1273, 1992.
- [73] R. Agarwal and C. Burrus, "Fast one-dimensional digital convolution by multidimensional techniques," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 22, pp. 1-10, 1974.
- [74] Xilinx. (2013). *System Generator for DSP user guides*, available:[http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_2/sysgen\\_user.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/sysgen_user.pdf).
- [75] V. Singh, A. Root, E. Hemphill, N. Shirazi, and J. Hwang, "Accelerating bit error rate testing using a system level design tool," in *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2003* pp. 62-68.
- [76] Xilinx. (2012). *Virtex-6 FPGA Family Overview*, Xilinx documentation , available:[http://www.xilinx.com/support/documentation/data\\_sheets/ds150.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf).
- [77] T. Tuan, A. Rahman, S. Das, S. Trimberger, and S. Kao, "A 90-nm Low-Power FPGA for Battery-Powered Applications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, pp. 296-300, 2007.
- [78] A. Yakovlev, "Energy-modulated computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE 2011)*, pp. 1-6.
- [79] M. B. Abdelhalim and A. E. Salama, "Implementation of 3D-DCT based video encoder/decoder system," in *International Symposium on Signals, Circuits and Systems, SCS 2003*. , pp. 389-392 vol.2.
- [80] A. H. Ghamarian, M. C. W. Geilen, S. Stuijk, T. Basten, A. J. M. Moonen, M. J. G. Bekooij, B. D. Theelen, and M. R. Mousavi, "Throughput Analysis of Synchronous Data Flow Graphs," in *Sixth International Conference on Application of Concurrency to System Design, ACSD 2006*, pp. 25-36.
- [81] A. Marshall and S. Boussakta, "Signal analysis of medical acoustic sounds with applications to chest medicine," *Journal of the Franklin Institute*, vol. 344, pp. 230-242, 2007.
- [82] S. W. Smith, *The Scientist & Engineer's Guide to Digital Signal Processing*: California Technical Publishing, 1997.

- [83] O. Nibouche, S. Boussakta, and M. Darnell, "A New Architecture For Radix-2 New Mersenne Number Transform," in *IEEE International Conference on Communications, ICC '06* pp. 3219-3222.
- [84] O. Nibouche, S. Boussakta, and M. Darnell, "Pipeline Architectures for Radix-2 New Mersenne Number Transform," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 56, pp. 1668-1680, 2009.
- [85] L. Litwin, "FIR and IIR digital filters," *Potentials, IEEE*, vol. 19, pp. 28-31, 2000.
- [86] K. S. a. J. K. Hammond, *Fundamentals of Signal Processing for Sound and Vibration Engineers*: John Wiley & Sons Ltd, 2008.
- [87] Z. Hussain, *Digital Image Processing: practical applications of parallel processing techniques*: Ellise Horwood Ltd, 1991.
- [88] E. Battenberg and R. Avizienis, "Implementing real-time partitioned convolution algorithms on convolutional operating systems " in *Proc. of the 14th Int. Conference on Digital Audio Effects (DAFx-11)*, Paris, France., 2011, pp. 1-8.
- [89] K. Suresh and T. V. Sreenivas, "Linear filtering in DCT IV/DST IV and MDCT/MDST domain," *Signal Processing*, vol. 89, pp. 1081-1089, 2009.
- [90] J. R. Mohammed and G. Singh, "Real-Time Implementation of New Adaptive Beamformer Sensor Array For Speech Enhancement in Hearing Aid," in *3rd International Conference on Intelligent Sensors, Sensor Networks and Information, ISSNIP 2007*, pp. 607-611.
- [91] G. Lightbody, R. Woods, and R. Walke, "Design of a parameterizable silicon intellectual property core for QR-based RLS filtering," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 11, pp. 659-678, 2003.
- [92] F. a. V. Wefers, M. , "Optimal filter partition for efficient convolution with short input/output delay," in *Proc. of the 14th Int. Conference on Digital Audio Effects (DAFx-11)*, 2011, pp. 155-161.
- [93] L. K. Patton and B. D. Rigling, "Autocorrelation Constraints in Radar Waveform Optimization for Detection," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 48, pp. 951-968, 2012.
- [94] R. S. Deepthi and S. Sankaraiah, "Implementation of mobile platform using Qt and OpenCV for image processing applications," in *IEEE Conference on Open Systems (ICOS)*, 2011, pp. 284-289.
- [95] J. Du, W. Zhang, S. Fu, P. Bian, X. Ning, and Z. Wang, "Inspection and Orientation Model of Digital Camera," in *Second International Conference on Information and Computing Science, ICIC '09*, 2009, pp. 61-64.
- [96] M. F. A. F. a. S.-C. H. Tong Hau Lee, "Intracranial Hemorrhage Annotation for CT Brain Images," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 1, pp. 2088-5334, 2011.
- [97] D. C. C. Wang, A. H. Vagnucci, and C. C. Li, "Digital image enhancement: A survey," *Computer Vision, Graphics, and Image Processing*, vol. 24, pp. 363-381, 1983.
- [98] L. Haifang, Q. Xiaoyan, C. Junjie, and X. Jie, "The study of data analysis methods based on fMRI brain-computer interface," in *IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2010)*, pp. 989-993.
- [99] A. T. Moreo, P. N. Lorente, F. S. Valles, J. S. Muro, and C. F. Andrés, "Experiences on developing computer vision hardware algorithms using Xilinx system generator," *Microprocessors and Microsystems*, vol. 29, pp. 411-419, 2005.

- [100] B. Attachoo and P. Pattanasethanon, "A new approach for colored satellite image enhancement," in *IEEE International Conference on Robotics and Biomimetics, ROBIO 2009*, pp. 1365-1370.
- [101] A. Ahmad, A. Amira, H. Rabah, and Y. Berviller, "Medical image denoising on field programmable gate array using finite Radon transform," *Signal Processing, IET*, vol. 6, pp. 862-870, 2012.
- [102] M. F. Bin Othman, N. Abdullah, and N. A. Bin Ahmad Rusli, "An overview of MRI brain classification using FPGA implementation," in *IEEE Symposium on Industrial Electronics & Applications (ISIEA 2010)* pp. 623-628.
- [103] Moctezuma, J. C. , S. nchez, R. and, and S. Ivarez, "Architecture for filtering images using Xilinx system generator," in *2nd WSEAS International Conference on Computer Engineering and Applications*, Acapulco, Mexico, 2008.
- [104] R. C. Decharms, "Applications of real-time fMRI," *Nature Reviews Neuroscience*, vol. 9, pp. 720-729, Sep 2008.
- [105] K. T. Gribbon, D. G. Bailey, and A. Bainbridge-Smith, "Development Issues in Using FPGAs for Image Processing," in *Proceedings of Image and Vision Computing* Hamilton, New Zealand, , 2007, pp. pp. 217–222.
- [106] C. T. G. Johnston, K. T. and Bailey, D. G., "Implementing Image Processing Algorithms on FPGAs," in *Proceedings of the Eleventh Electronics New Zealand Conference, ENZCon '04* Palmerston North, 2004, pp. 118-123.
- [107] A. Downton and D. Crookes, "Parallel architectures for image processing," *Electronics & Communication Engineering Journal*, vol. 10, pp. 139-151, 1998.
- [108] V. a. R. Elamaran, G., "FPGA implementation of point processes using Xilinx System Generator," *Journal of Theoretical and Applied Information Technology* vol. Vol. 41. , pp. pp 201 - 206, 2012.
- [109] H. F. Ladgham A, Sakly A, Mtibaa A (2012) Real Time Implementation of Detection of Bacteria in Microscopic Images Using System Generator, "Real Time Implementation of Detection of Bacteria in Microscopic Images Using System Generator," *J Biosens Bioelectron*, vol. 3, p. 127, 2012.
- [110] G. Papari and N. Petkov, "Edge and line oriented contour detection: State of the art," *Image and Vision Computing*, vol. 29, pp. 79-103, 2011.
- [111] L. Chang-song and J. Sheng-Zhen, "The Implement of High Speed Correlation Tracking Algorithm Based on FPGA in Space Solar Telescope," in *8th International Conference on Signal Processing* 2006.
- [112] I. Yasri, N. H. Hamid, and N. B. Z. Ali, "VLSI based edge detection hardware accelerator for real time video segmentation system," in *2012 4th International Conference on Intelligent and Advanced Systems (ICIAS) 2012*, pp. 719-724.
- [113] K. N. Kay, T. Naselaris, R. J. Prenger, and J. L. Gallant, "Identifying natural images from human brain activity," *Nature*, vol. 452, pp. 352-U7, Mar 20 2008.
- [114] R. Sitaram, N. Weiskopf, A. Caria, R. Veit, M. Erb, and N. Birbaumer, "fMRI Brain-Computer Interfaces," *Signal Processing Magazine, IEEE*, vol. 25, pp. 95-106, 2008.
- [115] N. Neumann and A. Kubler, "Training locked-in patients: a challenge for the use of brain-computer interfaces," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 11, pp. 169-172, 2003.
- [116] J. Fessler, "Model-Based Image Reconstruction for MRI," *Signal Processing Magazine, IEEE*, vol. 27, pp. 81-89, 2010.
- [117] J. A. Fessler, S. Lee, V. T. Olafsson, H. R. Shi, and D. C. Noll, "Toeplitz-Based Iterative Image Reconstruction for MRI With Correction for Magnetic Field Inhomogeneity," *Signal Processing, IEEE Transactions on*, vol. 53, pp. 3393-3402, 2005.



- [118] J. Song and Q. H. Liu, "An Efficient MR Image Reconstruction Method for Arbitrary K-space Trajectories Without Density Compensation," in *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS '06 2006*, pp. 3767-3770.
- [119] E. Corona, S. Mitra, and M. Wilson, "A fast algorithm for registration of individual frames and information recovery in fluorescein angiography video image analysis," in *Fifth IEEE Southwest Symposium on Image Analysis and Interpretation 2002*, pp. 265-268.
- [120] O. A. Elsayed, "Optimized algorithms for real time medical image registration," in *4th European Education and Research Conference (EDERC 2010)* pp. 121-124.
- [121] E. E. Konofagou, W. Manning, K. Kissinger, and S. D. Solomon, "Myocardial elastography - comparison to results using MR cardiac tagging," in *2003 IEEE Symposium on Ultrasonics* pp. 130-133 Vol.1.
- [122] Y. Ping, X. De, Z. Zhengtao, C. Guodong, and T. Min, "A vision system with multiple cameras designed for humanoid robots to play table tennis," in *2011 IEEE Conference on Automation Science and Engineering (CASE)*, pp. 737-742.
- [123] L. Jian, A. Xiangjing, Y. Lei, and H. Hangen, "A Reconfigurable Parallel Architecture for Image Computing," in *The Sixth World Congress on Intelligent Control and Automation, WCICA 2006* pp. 10491-10495.
- [124] Z. Hui, G. Zi, X. Mingxin, T. Zhiwei, and H. Guangshu, "A Reconfigurable System-on-chip Architecture for Medical Imaging: Preliminary Results," in *27th Annual International Conference of the Engineering in Medicine and Biology Society, IEEE-EMBS 2005*, pp. 1747-1749.
- [125] K. E. Jordan, D. A. Yuen, D. M. Reuteler, S. Zhang, and R. Haimes, "Parallel interactive visualization of 3D mantle convection," *Computational Science & Engineering, IEEE*, vol. 3, pp. 29-37, 1996.
- [126] V. Rajaravivarma, P. K. Rajan, and H. C. Reddy, "Design of multidimensional FIR digital filters using the symmetrical decomposition technique," *Signal Processing, IEEE Transactions on*, vol. 42, pp. 164-174, 1994.
- [127] E. Catmull and A. R. Smith, "3-D transformations of images in scanline order," *Comput Graphics (ACM)*, vol. 14, pp. 279-285, 1980.
- [128] H. Sungsoo, Z. Zhiyuan, K. Mueller, and S. Matej, "Efficiently GPU-accelerating long kernel convolutions in 3-D DIRECT TOF PET reconstruction via a kernel decomposition scheme," in *2010 IEEE Nuclear Science Symposium Conference Record (NSS/MIC)*, pp. 2866-2867.
- [129] D. Y. v. C. F. Kruggel, and X. Descombes, "Comparison of Filtering Methods for fMRI Datasets," *NeuroImage* vol. 10, pp. 530-543, 1999.
- [130] Y. Kobayashi, M. Hariyama, and M. Kameyama, "Optimal Periodic Memory Allocation for Image Processing With Multiple Windows," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, pp. 403-416, 2009.
- [131] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-Power Digital Signal Processing Using Approximate Adders," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, pp. 124-137, 2013.