RESEARCHERID
THOMSON REUTERS

[Parveen, 5(6): June 2018] ISSN 2348 – 8034
DOI10.5281/zenodo.1309261 Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

AN FPGA IMPLEMENTATION OF PARALLEL 2-D MRI IMAGE FILTERING ALGORITHM USING QUARTUS-II

Mohammad Sohana Parveen[1], Poonam Swami[2] & C.Deepika[3]
[*1]Assistant Professor, [2]Assistant Professor, [3]Assistant Professor
[1]Department of Electronics and Communication Engineering,
[1]KG Reddy College of Engineering & Technology, Hyderabad, India

ABSTRACT

In implementing parallel multi-dimensional image filtering algorithms, field programmable gate array (FPGA) provide beyond the low-level line-by-line hardware description language programming. High level abstract hardware-oriented parallel programming method can structurally bridge this gap. Currently, power is a major factor for implementing any algorithm. In this paper, image filtering algorithm is implemented on cyclone-IV FPGA device. By this, lower power consumption of 0.97W down to 0.39W respectively at maximum sampling frequency of up to 230 MHZ .the functional implementation of all processes using verilog HDL code of FPGA has been compiled on Quartus-II software tool.

Keywords: *image filtering algorithm, FPGA, Quartus-II.*

I. INTRODUCTION

In modern parallel algorithm applications such as image filtering , , DSP , medical imaging power consumption in portable image processing , satellite data processing high speed wavelet based image compress, MPEG-4 motion estimation in mobile applications and global communication link FPGAs are used .in most of the applications the solutions are FPGA-based are programmed with low-level hardware description languages (HDL) inherited from ASIC design methodologies.

Edge detection is a very complex process affected by deterioration due to different levels of noise. to solve the problem of edge detection a number of operators are defined. They are active for certain classes of images, but not suitable for others. Edge detection is a vital step in digital image processing. due to the limited processor speed , the image processing algorithms has been limited to software implementation which is slower. In order to increase the speed, the image processing algorithms are implemented on hardware. Because FPGA have added feature of parallelism, the edge detection can be effectively implemented.

The proposed FPGA implementation method is based on QUARTUS-II software tool. This method is tested on the efficient implementation of sobel 2-d image filtering algorithm, targeting cyclone-IV FPGA board.

Section-II describes the parallel 2-d image filtering algorithm, section-III describes the implementation On FPGA, section-IV gives the results and section-V describes the conclusion and future scope.

II. PARALLEL 2-D IMAGE FILTERING ALGORITHMS

Parallel 2-D MRI filtering algorithms are a 5x5 convolution kernel mask based image processing algorithms. Generally, the parallel architecture of these algorithms is constructed of serial to parallel input stage, 2-D convolution filtering vector for processing and a parallel to serial reconstructed output stage, as shown in below Fig

ISSN 2348 – 8034
Impact Factor- 5.070

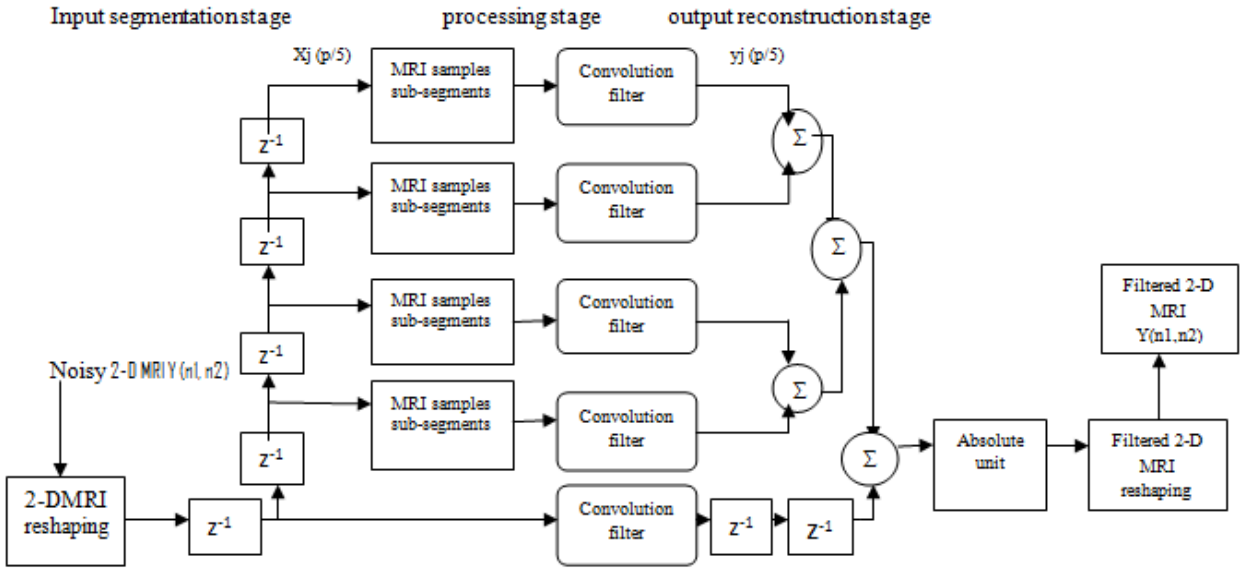


Fig - parallel 2-D MRI filtering algorithm

Input 2-D Segmentation MRI Stage

The serial to parallel input segmentation stage can be obtained by two steps. First step is reshaping. Second step is segmentation and buffering samples.

First step: the 2-D MRI matrix x (n1, n2) of size (N×N) is behaviorally reshaped, within the input stage, from (row × column) matrix to be (time stamp × MRIsamples) Matrix format. The reshaped MRI matrix has a time stamp in the first column and a vector containing the corresponding MRI samples stream in the subsequent column, x (t, p), as in (1) Eq. 1:

$$X(n1, n2) = X(t, p) \quad (1)$$

Where; t = 0, 1 … n1 × n2 -1 and p = 1, 2 … n1 × n2 Since the System Generator is a time based DSP development tool thus the time stamp variable, t in (1), is implicitly considered by the parallel MRI filtering algorithm. Hence (1) is simplified to Eq. 2:

$$X(n1, n2) = xn1, n2(p) = X(p) \quad (2)$$

Second step; the 2-D MRI samples stream, in (2), are equally split to five samples sub-segments.

Parallel 2-D convolution filtering stage

In The parallel 2-D filtering algorithm, using convolution filters vector the MRI pixel streams are processed as shown in Fig. Each convolution filter is a 5-tap MAC FIR filter. The filter architecture constructed using an image sample stream buffer, filter coefficient memory, comparator, address control unit, MAC unit and capture register.
The image sample stream buffer and the filter coefficient memory are used to store *N* MRI stream sub-segments and M coefficients respectively. The comparator generates the `reset' pulse and `enable' pulses for the accumulator and capture register respectively. when the address is zero the pulse is asserted and is delayed to account for pipeline stages. The address control unit gives the necessary address logic for the filter coefficient memory and the image sample stream buffer, in addition to the timing control for the comparator

The Convolution Filter algorithm is product of a set of M coefficients by N respective MRI samples subsequence to form an individual result. Each MAC FIR is characterized by its 1-D kernel, β (m1) of size (M), to convolve MRI samples sub-sequences, xj (p/5), of length N. This 1-D convolution filter produces filtered MRI samples sub-segment, yj (p/5).

As shown in Fig.1.1, five parallel MAC FIR filters, constitute a 5x5 filter which is characterized by its 2-D convolution kernel, β (m1, m2) of size (M × M).This 5x5 filter convolves five MRI samples subsequences, xj (p/5), of length N × N to produce a 2-D matrix filtered MRI samples sub-segment, yj (p).

Output 2-D MRI reconstruction stage

The final output 2-D MRI reconstruction stage is a parallel to serial conversion by summing up, pipelining and reshaping the filtered MRI samples sub-segments stream into the filtered 2-D MRI scan Since xm1, m2 (p) and Y n1, n2 (p) are to be a 2-D reshaped matrix for the MRI input, x (n1, n2) and a 2-D filtered MRI output, y(n1, n2), as shown in Fig.1.1 within the input stage and the output stage respectively.

III. IMPLEMENTATION ON FPGA

An architecture for the Gradient based edge detection algorithm using of Sobel operator is proposed and is implemented on an Cyclone IV Field Programmable Gate Array. In this process an image is taken as an input and it is converted into grayscale to obtain image intensity for edge detection. The Sobel edge detection operator is controlled by Finite State Machine (FSM) which executes a matrix area gradient operation to determine the level of variance through different pixels and to display the result on a monitor. The whole process is performed in the hardware level and implemented on Cyclone IV field programmable gate array platform.

This paper aims at designing the Gradient based edge detection algorithm using of Sobel operator in different methods and comparisons are made for these methods. In first method, the algorithm is written in C and simulated using Turbo C. In second method, the same algorithm is verified using Verilog. For simulations in this method the tool used is Model Sim XE III 6.21g. Finally the digital system design for Sobel Edge detection using Finite State machine is implemented on Cyclone IV FPGA.

At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations.

G_x= Im[i+2,j] + 2*Im[i+2,j+1] + Im[i+2,j+2] - Im[i,j] - 2*Im[i,j+1] - Im[i,j+2]

G_y= Im[i,j+2] + 2*Im[i+1,j+2] + Im[i+2,j+2] - Im[i,j] - 2*Im[i+1,j] - Im[i+2,j]

This operator places an emphasis on pixels that are closer to the center of the mask. The Sobel operator represents a rather inaccurate approximation of the image gradient, but is still of sufficient quality to be of practical use in many applications. More precisely, it uses intensity values only in a 3×3 region around each image point to approximate the corresponding image gradient, and it uses only integer values for the coefficients which weight the image intensities to produce the gradient approximation.

Procedure

1. Let Im [0…N, 0…M] containing original image
2. The magnitude is calculated using column gradient and row gradient
3. The out put image is a new image newIm [0…N-2, 0…M-2] which contains edge values.
4. Reads the original image, applies gradient derivative on the convolution basis, and obtains a new value, for central pixel, and places it into the new image.

In the second method architecture of Sobel Edge Detection algorithm design verified using Verilog,

[Parveen, **5(6): June 2018]**
DOI10.5281/zenodo.1309261
ISSN 2348 – 8034
Impact Factor- 5.070

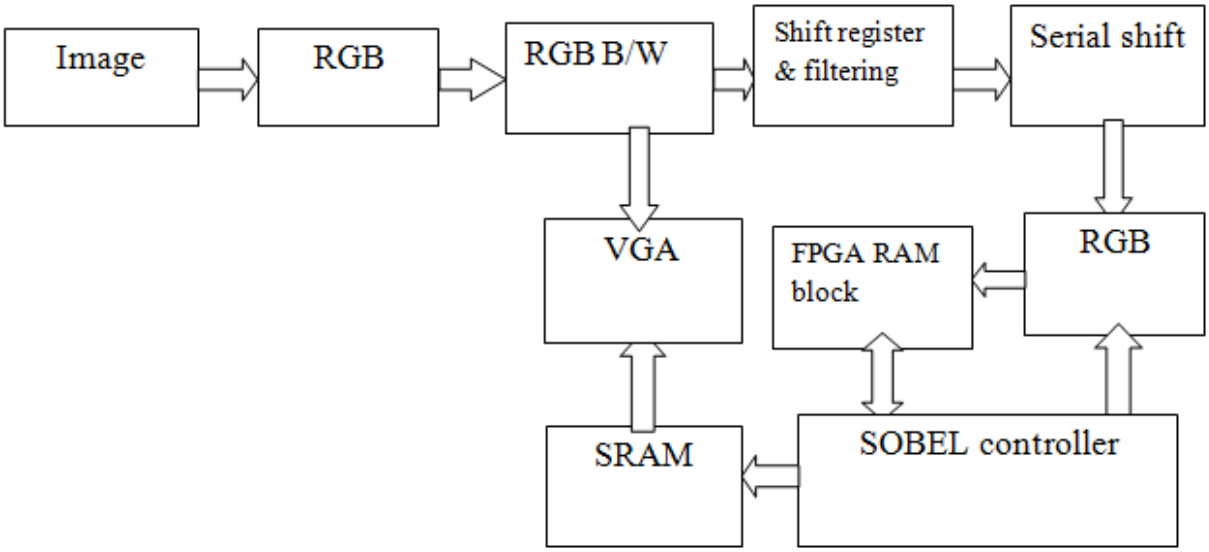


Fig - Block diagram of sobel edge detection and filtering

Initially an image is taken as input for the process. This image is read by using MATLABR2008a. Then the RGB b/w converter converts this RGB color space image into a black and white color space. After that the data will be transferred into serial data of image pixel which is carried out by shift register controller. Then serial shift register block would receive the emerged output of the previous stag. Afterwards, this image pixel data is stored in the SDRAM block according to the data address. Further to implement the design FPGA RAM block is taken as the target. Automatically activation of Sobel algorithm can be happened in this block immediately after the download of the design.

To perform the convolution process Sobel controller block controls the data flow from SDRAM to FPGA RAM blocks and updates the results to FPGA RAM block for next convolution .Finally, the result would be send to SRAM. This SRAM block stores the image by detecting its edge with the help of sobel algorithm. Image data transformation into VGA display format is controlled by the VGA controller block. Ultimately the image was displayed by the VGA display block with its detected edge

The hardware implementation on Cyclone IV Field Programmable Gate Array requires raw pixel information. Therefore, here convert a standard image format (such as JPEG, PNG, and BMP) to a raw image. In this format, use 8 bits to represent a pixel, i.e., 00000000 represents a pixel which is completely black, while 11111111 represents a pixel which is completely white. The extracted image is represented as a 2-dimensional array of integer values ranging from 0 to 255 corresponding to the individual pixels of the image. For colour images the raw data requires three 8-bit data corresponding to each of the primary colours Red, Green and Blue and will have to be processed one by one in the Cyclone IV FPGA.

After that the data would be transferred into serial data of image pixel which is carried out by shift register controller. Then serial shift register block would receive the emerged output of the previous stage. Afterwards, this image pixel data is stored in the SDRAM block according to the data address.

Since the camera result is outputted to the SDRAM fifo, we need to load the image into three Ram block (M4K) block sequentially as follows. The M4K blocks takes in 3 rows of pixels for edge detection computation

ISSN 2348 – 8034
Impact Factor- 5.070

1st M4K	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	…	a717	a718	a719	a720
2nd M4K	b1	Xx	xx	xx	xx	xx	xx	xx	Xx	xx	…	xx	xx	Xx	xx
3rd M4K	xx	Xx	xx	xx	xx	xx	xx	xx	Xx	xx	…	xx	xx	Xx	xx

Figure (1)

After three rows of pixels are inputted, edge detection state machine will begin to start computation on the first 3x3, as bolded below

1st M4K	**a1**	**a2**	**a3**	a4	a5	a6	a7	a8	a9	a10	…	a717	a718	a719	a720
2nd M4K	**b1**	**b2**	**b3**	b4	b5	b6	b7	b8	b9	b10	…	b717	b718	b719	b720
3rd M4K	**c1**	**c2**	**c3**	c4	c5	c6	c7	c8	c9	c10	…	c717	c718	c719	c720

Figure (2)

The edge detection result is then output to SRAM for VGA output. In the next iteration, the values are shifted to allow for next edge detection

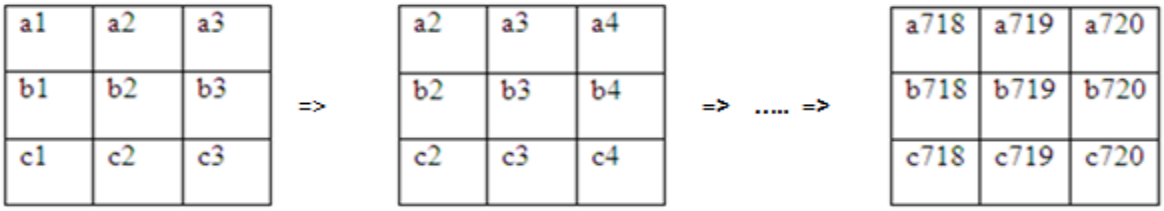


Figure (3)

After one row of computation, we read one more row of M4K blocks. This time, we only need to read in row 4, and reusing the values of row 2 in the 2 nd M4K, and row 3 in the 3 rd M4K.

1st M4K	**d1**	**d2**	**d3**	**d4**	**d5**	**d6**	**d7**	**d8**	**d9**	**d10**	**…**	**d717**	**d718**	**d719**	**d720**
2nd M4K	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	…	b717	b718	b719	b720
3rd M4K	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	…	c717	c718	c719	c720

Figure (4)

Extra care is taken in the edge detection state machine to ensure that we are doing computation with the correct data, so we are using the values b1, c1, d1 in the right order, and not d1, b1 and c1. In the next iteration, it becomes the following:

1st M4K	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	…	d717	d718	d719	d720
2nd M4K	**e1**	**e2**	**e3**	**e4**	**e5**	**e6**	**e7**	**e8**	**e9**	**e10**	**…**	**e717**	**e718**	**e719**	**e720**
3rd M4K	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	…	c717	c718	c719	c720

Figure (5)

And then

1st M4K	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	…	d717	d718	d719	d720
2nd M4K	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	…	e717	e718	e719	e720
3rd M4K	**f1**	**f2**	**f3**	**f4**	**f5**	**f6**	**f7**	**f8**	**f9**	**f10**	**…**	**f717**	**f718**	**f719**	**f720**

Figure (6)

The pattern repeats itself until the entire image is computed. It then continues to compute the next frame.

Further to implement the design FPAG RAM block is taken as the target. Automatically activation of Finite state machine of sobel controller can be happened in this block immediately after the download of the design. To perform the convolution process sobel controller block controls the data flow from SDRAM to FPGA RAM blocks and updates the results to FPGA RAM block for next convolution.

Finite State Machines

Here the Finite State Machine of sobel controller builded by 14 states. Among those states two state machines are implemented in this paper. The first state machine is responsible for reading the captured.

image from SDRAM and a second state machine is responsible for the edge detection algorithm. Each state performance an unique task. These states run certain operations which are constrained by certain time period. To change from one state to next state they must fulfill some conditions .The relationship between one state with the other state has showed by direction of arrow. Table 1 represents the changing conditions of states in detail

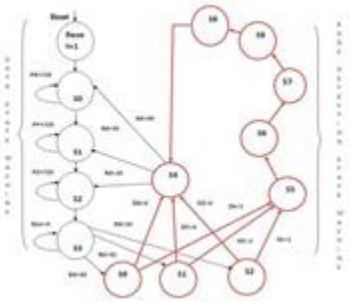


Fig - Finite state machine of sobel controller

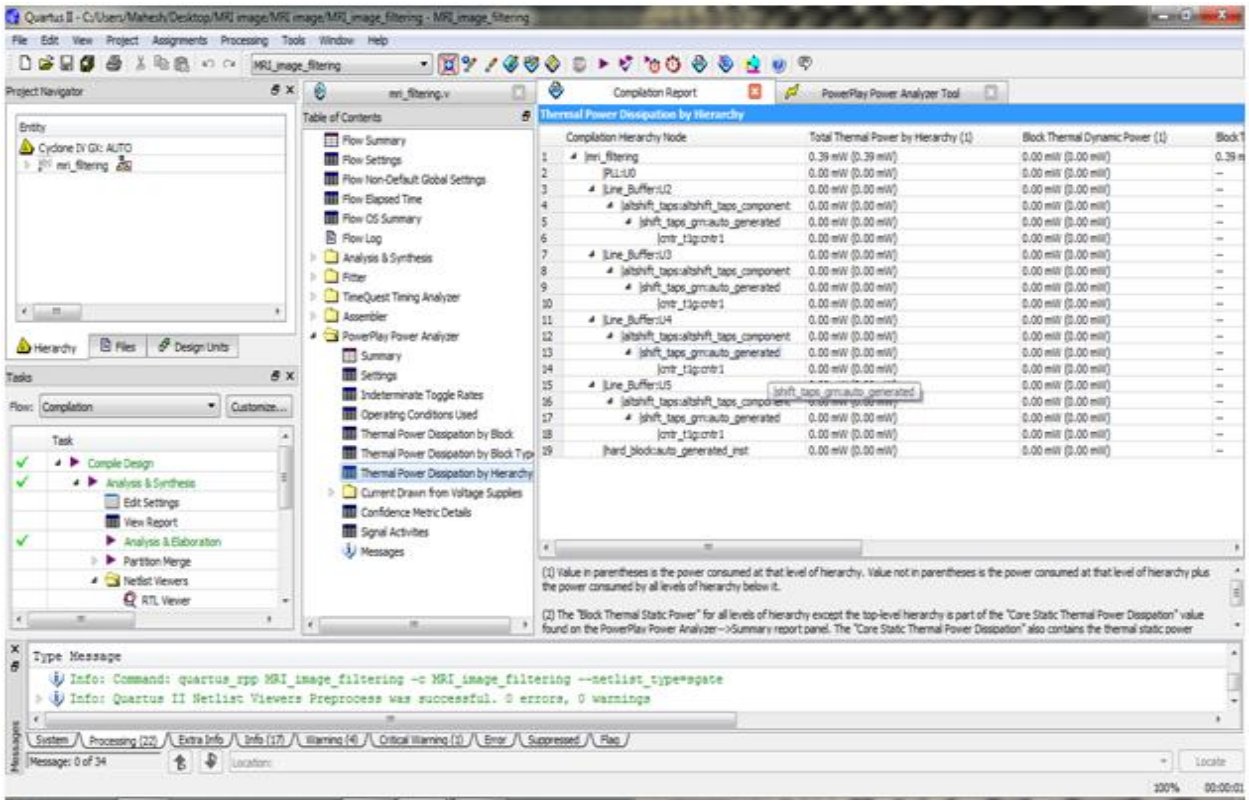
Table 1- description of states in finite state machine

State	Remark	Condition
	Idle	Reset
S0	Write 1st 720 pixels in 1st FPGA RAM block	RESET=0,E1=1,PX=720
S1	Write 1st 720 pixels in 1st FPGA RAM block	PY=720,E3=1
S2	Write 1st 720 pixels in 1st FPGA RAM block	PZ=720,E3=1
S3	Run edge	E1=0,E2=0,E3=0
S0	Check 3rd FPGA RAM block fulfill read 2 pixels	Read =00
S1	Check 3rd FPGA RAM block fulfill read 2 pixels	Read=01
S2	Check 3rd FPGA RAM block fulfill read 2 pixels	Read =10
S5	Calculate 8 directions and 8 to 4 edge detection directions or edge detection gradient calculation	DX=1,DY=1,DZ=1
S6	Absolute value	Sobel operator
S7	Direction calculation	Sobel operator
S8	Max direction calculation	Sobel operator
S9	Check edge versus threshold and display	Disp =1, Sobel operator
S4	Check whether edge calculation has reached 720	Disp =0

Finally, the result would be send to SRAM. This SRAM block stores the image by detecting its edge with the help of sobel algorithm. Image data transformation into VGA display format is controlled by the VGA controller block. Ultimately the image was displayed by the VGA display block with its detected edge.

ISSN 2348 – 8034
Impact Factor- 5.070

IV. RESULT



V. CONCLUSION

To provide fast FPGA prototyping for high performance computation, new FPGA implementation methods developed in this paper. The FPGA implementation is behaviorally targeted to cyclone-IV FPGA board using the Quartus II 10.1 software tool. by using this methodology ,which is high-level abstract hardware-oriented parallel programming, to outperform the low-level line-by-line HDL programming, with excellent quality for parallel 2-D MRI image filtering algorithms of power consumption down to (0.39) at maximum frequency of up to (230 MHz)

VI. FUTURE SCOPE

The future work will be focused on the high performance efficient FPGA implementation for the parallel 3-D image filtering algorithms of the next generation advanced DSP applications within aerospace, defence, digital communications, multimedia, video and imaging industries.

REFERENCES

1. *M. Kiran, K. M. War, L. M. Kuan, L. K. Meng and L.W. Kin,"Implementing image processing algorithms using 'Hardware inthe loop' approach for Xilinx FPGA," Electronic Design, ICED2008, International Conference, Dec. 2008, PP.:1 – 6.*
2. *W. Atabany and P. Degenaar, "Parallelism to reduce power consumption on FPGA Spatiotemporal image processing," Proc. IEEE International Symposium on Circuits and Systems, ISCAS 2008, pp. 1476—1479.*
3. *R. Gao, D. Xu and J.P. Bentley, "Reconfigurable Hardware Implementation of an Improved Parallel Architecture for MPEG- 4 Motion Estimation in Mobile Applications," IEEE Transactions on Consumer Electronics, Vol. 49, 2003, pp.: 1383- 1390.*

4. *K. R. Nataraj, S. Ramachandran and B. S. Nagabushan, "Development of Algorithm, Architecture and FPGA Implementation of Demodulator for Processing Satellite Data*
5. *Communication" IJCSNS International Journal of Computer Science and Network security, 2009, VOL.9, pp.:137-147.*
6. *O. Nibouche, S. Boussakta and M. Darnell,"Pipeline architectures for radix-2 new Mersenne number transform ," IEEE Transactions on Circuits and Systems I: Regular Papers 56 (8), 2009, pp. 1668-1680.*
7. *O. Nibouche, S. Boussakta and M. Darnell, " A new architecture for radix-2 new Mersenne number transform," IEEE International Conference on Communications 2006, pp. 3219- 3222.*
8. *A. Masoudnia, H. Sarbazi-Azad and S. Boussakta, "Design and performance of a pixel - level pipelined - parallel architecture for high speed wavelet-based image compression" Computers and Electrical Engineering 31 (8), 2005, pp. 572-588.*
9. *T. Mak, et al, "Implementation of wave-pipelined interconnects inFPGAs," Proceedings - Second IEEE International Symposiumon NOCS 2008, , pp. 213-214.*
10. *C. Chang, "Design and application of a reconfigurable computing System for High Performance Digital Signal Processing", Ph.D. thesis, University of California, Berkeley, 2005.*
11. *S. Boussakta, "A novel method for parallel image Processing applications," Journal of Systems, 45 (10), 1999, pp. 825-839*
12. *Aziz, M., 2004. Parallel Digital Filtering Algorithms for Multiprocessor DSP systems. A PhD Thesis, University Of Leeds.*
13. *R. Woods, J. McAllister, G. Light body and Y. Yi, "FPGA-based Implementation of Signal Processing Systems" 2008, John Wiley &Sons, Ltd.*
14. *Hasan, S., A. Yakovlev and S. Boussakta, 2010. Performance efficient FPGA implementation of parallel 2-D MRI image filtering algorithms using Xilinx system generator. Proceedings of the 7thInternational Symposium on Communication Systems Networks and Digital Signal Processing,Jul. 21-23, IEEE Xplore Press, Newcastle UponTyne, pp: 765-769.*