Military Technical College
Kobry El-Kobbah,
Cairo, Egypt

7th International Conference
on Electrical Engineering
ICEENG 2010

General Structure Design and simulation for Image Processing

By

Wael Wasfy* **Hong Zheng**** **Wu Xinghua***** **Li Jun******
wwassfy@yahoo.com julyanna@vip.sina.com

Abstract:

Designing a general structure for fast image processing algorithms to compute the image intensity for low level 3x3 algorithms having the same parallel calculation method but with different kernel is our goal in this paper. Using DSP slice module inside the FPGA was an objective task to get the advantage of it as a faster, accurate, higher number of bits in calculations and different calculated equation maneuver capabilities. Using the remade design scheme can solve many problems such as redesign time consumption, increasing the calculation accuracy by using high number of bits in calculations more than the regular logic gates, saving FPGA resources is one of the advantages we got by using the general structure design especially if we need to make more than 3x3 low level algorithm for the same project we can select which algorithm to be calculated at certain time for the same structure as we take an example in this paper by selecting between Gaussian filter and Sobelx edge detector as an example. The General structure is capable to be modified to add other algorithms which have the same calculation methods. Other designers as mentioned later an example on this paper uses separate fixed design for each algorithm with maximum 12bit calculation accuracy.

Keywords:

Image processing; FPGA; Embedded processor

---

*
**
***
****
School of Automation Science and Electrical Engineering,
Beijing University of Aeronautics and Astronautics,
Beijing, 100191, P. R. China

1. Introduction:

For achieving the Design of the general structure for low level fast image processing we found the most common way in hardware design are ASIC, DSP and FPGA. Our goal was to improve the performance and the accuracy with as minimum as possible hardware resources and cost effective, so we tried to study the hardware design implementation possibilities.

If one decides to utilize an embedded system for a computer vision application, there is currently the choice between either using Digital Signal Processors (DSPs) or Field Programmable Gate Arrays (FPGAs) from different vendors. The design considerations for FPGAs are wide multiplication units, numerous logic elements, parallel hardware structures, handling of high data rates and the reconfigure-ability of FPGAs. Compared to high end DSPs, FPGAs are more expensive, the design and development of FPGA algorithms require more time and the processing power for sequential computations is slower than on DSPs because of the higher clock frequency of DSPs. General purpose DSPs are designed to fit a variety of market applications, with no consideration for any special algorithm [10]

There are two types of technologies available for hardware design. Full custom hardware design also called as Application Specific Integrated Circuits (ASIC) and semi custom hardware device, which are programmable devices like Digital signal processors (DSPs) or Field Programmable Gate Arrays (FPGAs).

ASIC design offers highest performance, but the complexity and the cost associated with the design is very high. The ASIC design cannot be changed and the design time is also very high. ASIC designs are used in high volume commercial applications. In addition, during design fabrication the presence of a single error renders the chip useless.

DSP is a class of hardware devices that fall somewhere between an ASIC and a PC in terms of the performance and the design complexity. DSPs are specialized microprocessors, typically programmed in C, or with assembly code for improved performance. It is well suited to extremely complex math intensive tasks such as image processing. Knowledge of hardware design is still required, but the learning curve is much lower than other design choices [1].

FPGA ia mainly used for computationally demanding functions like convolution filters, motion estimators, two-dimensional Discrete Cosine Transforms (2D DCTs) and Fast Fourier Transforms (FFTs) all are better optimized when targeted on FPGAs [6,7].

In an application that requires real-time processing, like video or television signal processing or real-time trajectory generation of a robotic manipulator, the specifications are very strict and are better met when implemented in hardware [3-5]. Features like embedded hardware multipliers, increased number of memory blocks and system-on-a-chip integration enable video applications in FPGAs that can outperform conventional DSP designs [2,8].

Design Gaussian filter and Sobelx algorithms low level algorithms have been done before primarily by using the logical unit components. Once the algorithm is different, the structure needs to be redesigned and the time and precision adjustment will be required to deal with a major adjustment in the structure. From this drawback we tried In our design to avoid redesign time consumption by making a general design core to be used now for two algorithms and can be modified later with and keeps the main design core for using it with other algorithms, also we used DSP slice module inside the FPGA to gain 48 bit in Addition and 18 bit by 18 bit in Multiplication which improves the accuracy in calculations as the other design used maximum 12bit in both calculations (adder and multiplier).

We did our best to conclude all of the above best features in our design due to parallelism in these image processing algorithms, and because of its different function each has its own method of calculation, so we take advantage of FPGAs internal DSP module as a unit structure to build our parallel processing array design. trying to gain the maximum advantages as much as we used FPGA to get low cost compared to ASIC and using inside the DSP slice module to get a higher speed and increases number of bits for improving calculations accuracy, all will be implemented in hardware for better performance stability compared to software designs, also using another design as a guide to compare our design, helped us a lot, then we proved and verify our design for both algorithms.

2. General structure of image processing algorithms

2.1. Low-level Vision Algorithms

- ***Gaussian pyramid [4,9]***

Two dimensional low-pass filters, such as the Gaussian low-pass filter, work with a filter kernel, calculate an average value for a destination pixel using a number of neighboring source pixels. The two dimensional Gaussian filter is shown in following figure 1.

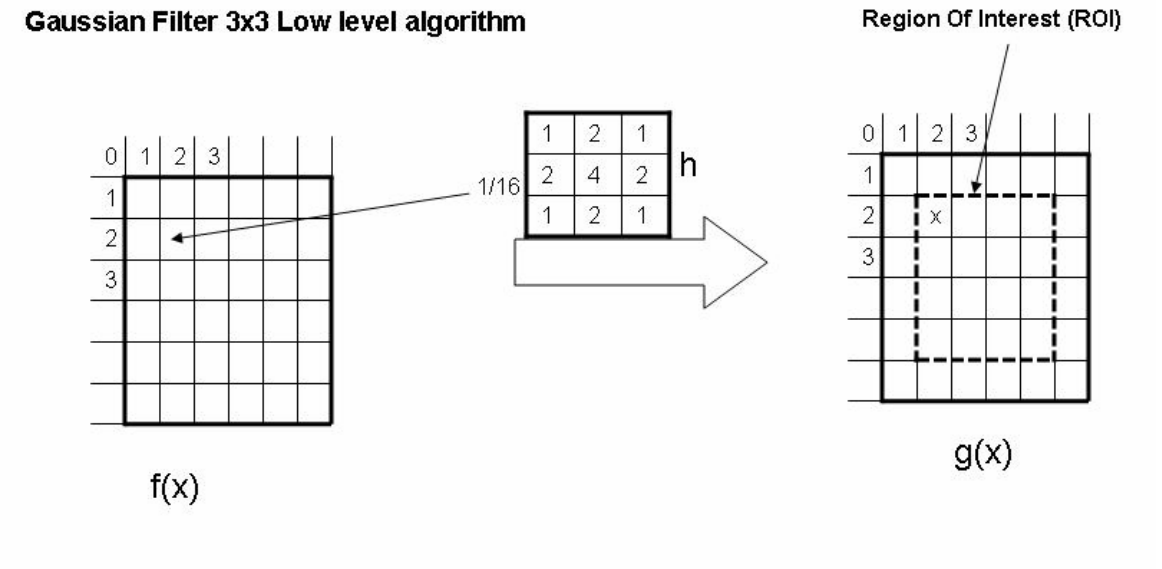


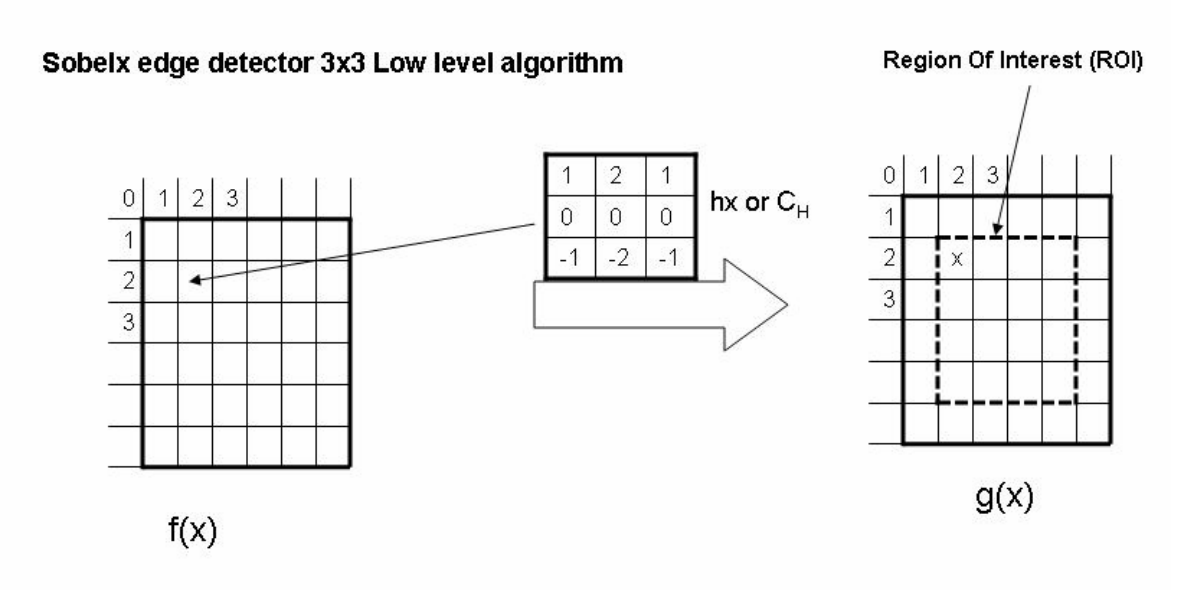
Figure (1): *Gaussian pyramid filter kernel*

When dealing with digital images integer weighting factors are used. A typical 3x3 Gaussian filter matrix and the decimation of the pixels are shown in Figure 1. The anchor point of the Gaussian filter kernel is marked with an"X".

Obviously for every calculated pixel one neighboring pixels in both dimensions are required. Therefore, this function uses a Region Of Interest (ROI). With every Gaussian pyramid level the number of pixels in x- and y-coordinates is reduced by a factor of 1.

- ***Sobel edge detector [4,9]:***

The Sobelx edge detector performs a gradient measurement over the x- and y-coordinates with separate filter kernels for each dimension, see Equation 2. Regions with high spatial frequency correspond to edges. Typically the Sobel filter kernel consists of a pair of 3x3 coefficients as shown in Figure 2 and is designed in such a way to have a maximum response for edges running vertically and horizontally through the image.



Figure(2): *Sobelx edge detector*

According to Gaussian and Sobel similarities on their calculation method for using the neighbor pixels window 3x3 to get the new pixel data, taking into consideration that their differences in their kernel's. Taking into consideration that; expressing the kernel operation in each frame process can be decomposed into block processing mode, and this block has the same processing function and the number and size determine degree of algorithm parallelism.

The highest degree of parallelism for one (640x480) image frame is 640x3 delay element blocks with. Computing time is determined by number of machine cycle to get the first frame pixel calculated plus number of frame pixel remains which it will be equal to the machine cycles without other delays = 9 + 640*480= 307209 machine cycle for working in frequency 100MHz (10nSec) for comparison with other systems we found that time = 10nSec * 307209 = 3.07209 mSec almost 3.07mSec for single frame. so from that; we defined a new general structure design of fast image processing as we will explain later

3. Design General structure of FPGA

3.1. There are three ways to program a DSP48 in System Generator:

- **Use Standard Components**

Map designs to Mult and AddSub blocks or use higher level IP such as the MACFIR filter generator blocks. This approach is useful if the design needs to be compatible with V2P or S3 devices or uses a lower-speed clock and the mapping to DSP48s is not required.

- **Use Synthesizable Blocks**

Structure the design to map onto the DSP48's internal architecture and compose the design from synthesizable Mult, AddSub, Mux and Delay blocks. This approach relies on logic synthesis to infer DSP48 blocks where appropriate. This approach gives the compiler the most freedom and can often achieve full-rate performance.

- **Use DSP48 Blocks**

Use System Generator's DSP48 and DSP48 Macro blocks to directly implement DSP48-based designs. This is the highest performance design technique. Be aware however that obtaining maximum performance and minimum area for designs using DSP48s may require careful mapping of the target algorithm to the DSP48's internal architecture, as well as the physical planning of the design [15].

3.2. Delay line elements

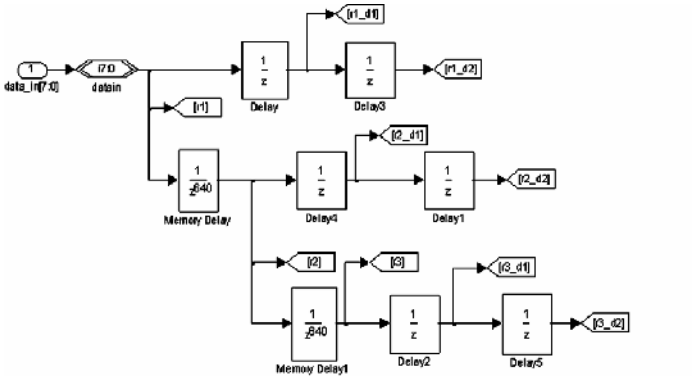


Figure (3): *Line buffer principle for a 2D FIR filter for a 640x480 frame.*

Incoming pixels are processed by means of a simple 2D digital Finite Impulse Response (FIR) filter convolution kernel, working on the grayscale intensities of each pixel's neighbors in a 3x3 region. Image lines are buffered through delay-lines producing primitive 3x3 cells where the filter kernel applies. The line-buffering principle is shown in Figure 3. A z -1 delay block produces a neighboring pixel in the same scan line, while a z-640 delay blocks produces the neighboring pixel in the previous image scan line. We assume image size of 640x480 pixels.

The line-buffer circuit is implemented in the same manner for both noise and edge filters. Frame resolution is incorporated in the line buffer diagram as hardware built in parameter. If a change in frame size is required we need to redesign and recompile. The number of delay blocks depends on the size of the convolution kernel, while delay line depth depends on the number of pixels in each line. Each incoming pixel is at the center of the mask and the line buffers produce the neighboring pixels in adjacent rows and columns. Delay lines with considerable depth are implemented as dedicated RAM blocks in the FPGA chip and do not consume logical elements [18].

3.3. Design Using DSP48 Macro Blocks

This block is listed in the following Xilinx Blockset libraries: Index, Control. The System Generator DSP48 Macro block provides a device independent abstraction of the blocks DSP48, DSP48A, and DSP48E. Using this block instead of using a technology-specific DSP slice helps makes the design more portable between Xilinx technologies. Depending on the target technology specified at compile time, the block wraps one DSP48/DSP48A/DSP48E block along with reinterpret and convert blocks for data type alignment, multiplexers to handle multiple opmodes and inputs, and registers [15].

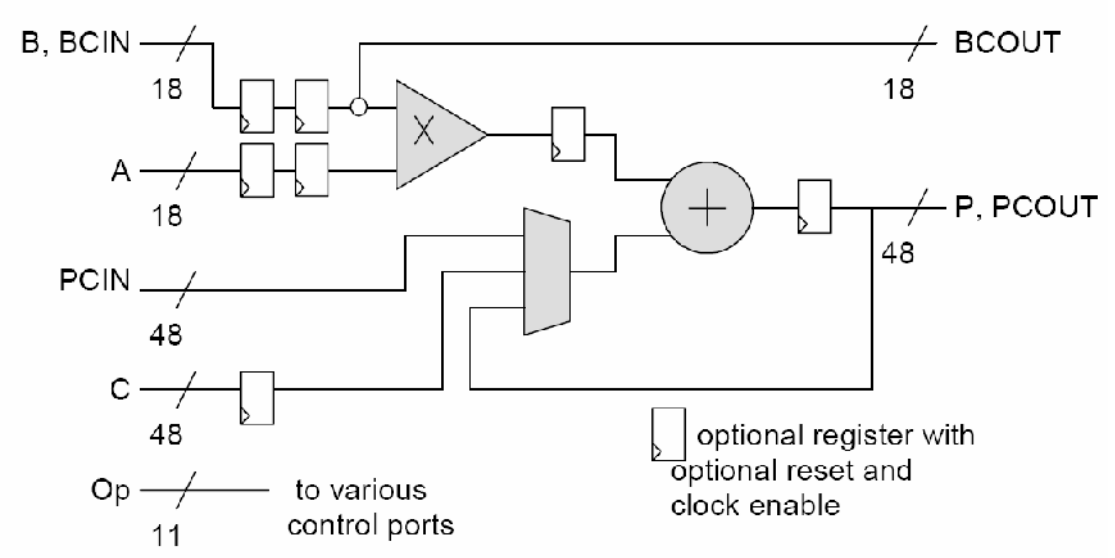


Figure (4): *DSP48 slice in Xilinx Virtex 4 FPGA*

Consider the simple model using DSP48 Macro which has three inputs defined as Xo, Yo, and Zo. Because more than one Instruction opmode can be specified in block dialog box, the Sel input port is automatically added:

In our design two legal opmodes are entered in the Instructions field.

When 0 is specified on the Sel input, the first instruction opmode is implemented. The value needed will be to implement the Gaussian 3x3 2D FIR filter for all DSP blocks directly to output P.

When 1 is specified on Sel, the second Instruction opmode will be implemented. In this case, Sobel x edge detector will appear at the output.

When this design is compiled, if the target technology is Virtex-4, then a DSP48 slice will be netlisted. If Virtex-5 is specified, then a DSP48E slice will be netlisted, and if the Spartan-3A DSP technology is specified, then a DSP48A slice will be used in the implementation [17].

3.4. Proposed design using DSP's for 2D Filter Image processing

A Simulink blockset is a library of blocks that can be connected in the Simulink block editor to create functional models of a dynamical system. For system modeling, System Generator blocksets are used like other Simulink blocksets.

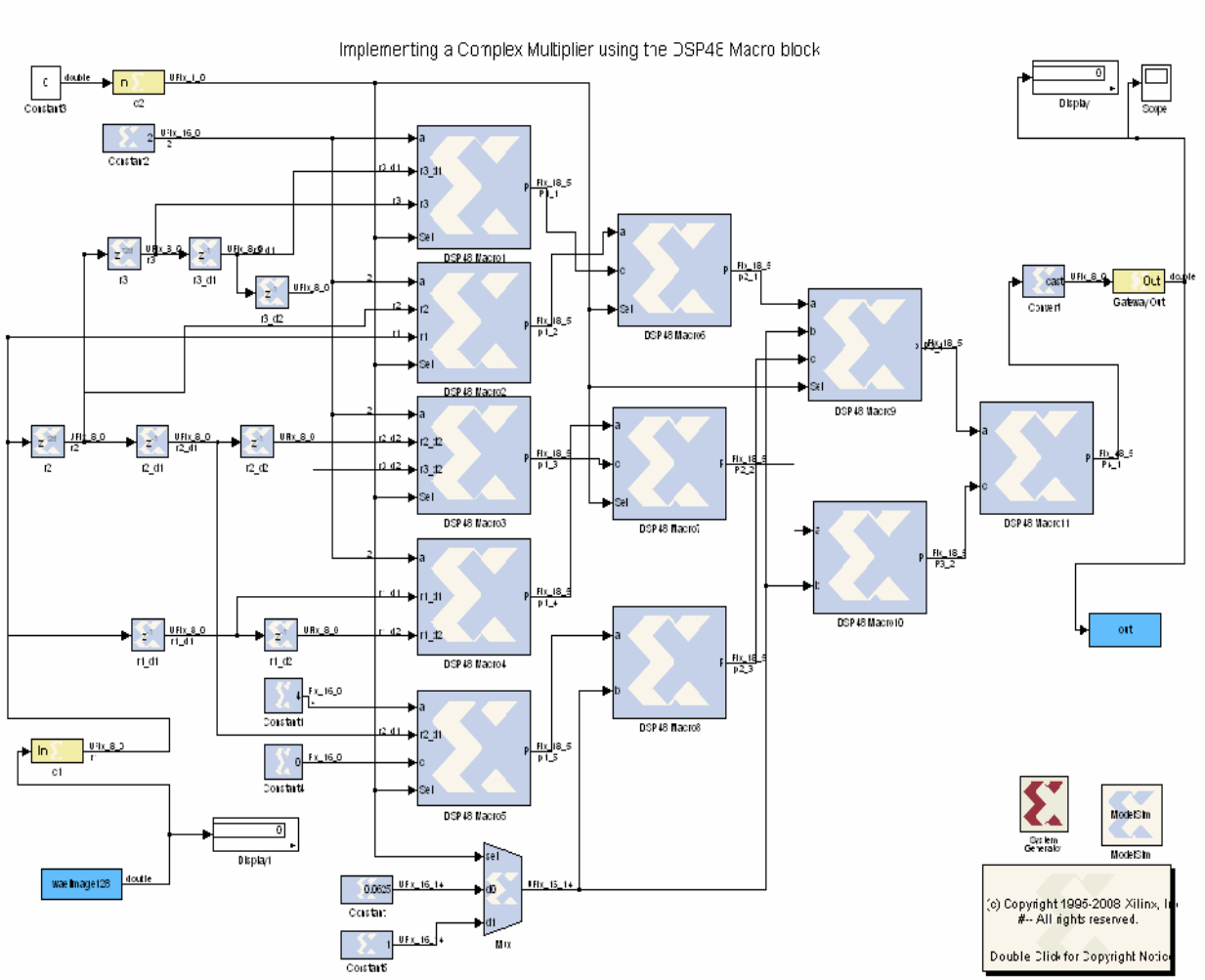


Figure (5): *Proposed structure for Gaussian / Sobelx*

In our design we used DSP slice which is 48 bit Adder and 18 bit by 18 bit Multiplier which gives high accuracy in calculations associated with high number of bits used. Also each module can work as adder or multiplier or adder and multiplier at the same time. The other and great benefit that pre-implemented equations can be saved in each module and can be selected accordingly to select pin status which allows different equation respective to different algorithm.

There are also blocks that provide interfaces to other software tools (e.g., FDATool, ModelSim) as well as the System Generator code generation software [15].

Proposal of using DSP48 instead of using standard RTL components

1	2	1
2	4	2
1	2	1

Gaussian filter weights

1	2	1
0	0	0
-1	-2	-1

Sobelx Edge detector weights

R1_d2	R1_d1	R1
R2_d2	R2_d1	R2
R3_d2	R3_d1	R3

Signal variable description

Define the Constants multiplied in our 9 inputs

For calculating one pixel data value in both algorithms we need 9 input neighbor pixel multiplied by its algorithm weight value. According to different algorithms weights values we will consider it as variables (R1, R2, R3, R1_d1, R2_d1, R3_d1, R1_d2, R2_d2, R3_d2) each variable in the 3x3 matrix correspond to two different weight values and these values will be selected on the general structure design due to selecting pin "C1". Also we can use the equality of the weight values in Gaussian filter for simplification the design by using one multiplier constant as:

- R1 = R3 = R1_d2 = R3_d2 = 1
- R2 = R1_d1 = R2_d2 =R3_d1 = 2
- R2_d1 = 4

In Sobelx for design simplification we will use the similar weights value but with negative sign in the third row, also second row weights are all zeroes as all will be cleared in the equations and design figure.

The signal flow are divided into levels of calculations, these levels defined as each electronic element will cause delay in the signal calculation flow will be consider as a level, the delay line elements are an obligation for certain frame size and it will be taking into consideration one time for calculating the first pixel data as an overall one time delay, then data will flows consequently. The following listed equations are describing the equations will be calculated in each DSP level for different algorithm and its corresponding algorithm weight variables.

Equations for image data flows according to select pin C1 (0 or 1)

For more clarification of using the DSP module capability of choosing two different equations working in the same design works together, on the following table 1 we replaced all DSP's module by its related calculated equation.
On the table cells there are two different equations for the same DSP module inputs (A, B, and C) and same output (P) with a control pin (SEL) to determine which equation will be activated at certain SEL data value

	First level	**Second level**	**Third level**	**Fourth level**
Input A Input B	Case SEL =0 P1_1 = C + A * B			
Input C SELECT	Case SEL = 1 P1_1 = C + A * B	Case SEL =0 *P2_1 = C + A*		
Input A Input B	Case SEL =0 *P1_2 = C + A * B*	Case SEL = 1 *P2_1 = C – A*	Case SEL =0 *P3_1 = C + A * B*	
Input C SELECT	Case SEL = 1 *P1_2 = C*		Case SEL = 1 *P3_1 = A * B'* Note 1	
Input A Input B	Case SEL =0 *P1_3 = C + A * B*	Case SEL =0 *P2_2 = C + A*		Case SEL =0 P4_1 = C + A
Input C SELECT	Case SEL = 1 *P1_3 = C*	Case SEL = 1 *P2_2 = C – A*	Case SEL =0 P3_2 = A * B	Case SEL = 1 P4_1 = C + A
Input A Input B	Case SEL =0 P1_4 = C + A * B		Case SEL = 1 P3_2 = A * B' Note 1	
Input C SELECT	Case SEL = 1 P1_4 = C + A * B	Case SEL =0 P2_3 = A * B		
Input A Input B	Case SEL =0 *P1_5 = A * B*	Case SEL = 1 P2_3 = A * B' Note 1		
Input C SELECT	Case SEL = 1 *P1_5 = C*			

***Table (1):** DSP equation representation*

Note 1:
Selecting (SEL = C2= "0") for Gaussian algorithm B = 0.0625 and for
Selecting (SEL = C2= "1") for SobelX Algorithm B = 1

4. Evaluation of the system performance and comparison with other systems

- ***System performance***

Integrated Software Environment (ISE 10.1) software for FPGA and Matlab mathworks (R2007a) software both were used for design, validate and simulate our general structure fast image design, the FPGA embedded system used is Xilinx Company product Virtex4, XC4VSX55-12FF1148, explaining our implementation design overall resources of the FPGA capabilities will be appeared in Table 2 according to ISE utilities.

For the same architecture and the proposed FPGA Design input / output description as descriped in Table 3, and during calculating the two different algorithms according to the selected pin (C2) we have seen that how many machine cycle needed for the first pixel output data calculated from the Serial Image input data. It appeared that it needed 9 machine cycles (90 nSec for 100MHz frequency as described in figure 6). Also we had taken into consideration releasing the Direct Clock Manager (DCM) signal which allows the clocking system to take place in FPGA, the input serial image data will continue flows through pin (C1) as a result output flows through pin (gateway).

XILINX VIRTEX 4 Number XC4VSX55-12FF1148				
Logic utilization	**Used**	**Available**	**Utilization**	**Notes**
Number of slice Flip Flops	664	49,152	1%	
Number of 4 inputs LUTs	629	49,152	1%	
Logic distribution				
Number of occupied slices	639	24,576	2%	
	639	639	100%	
	0	639	0%	
Total number of 4 input LUTs	**629**	**49,152**	**1%**	
Number used as logic	21			
Number used as shift registers	608			
Number of bonded IOBs	21	640	3%	
Number of BUFG/BUFGCTRLs	2	32	6%	
Number used as BUFG	1			
Number used as BUFGCTRLs	1			
Number of DSP48s	11	512	2%	
Number of DCM_ADVs	1	8	12%	

Table (2): *FPGA Utilization resources*

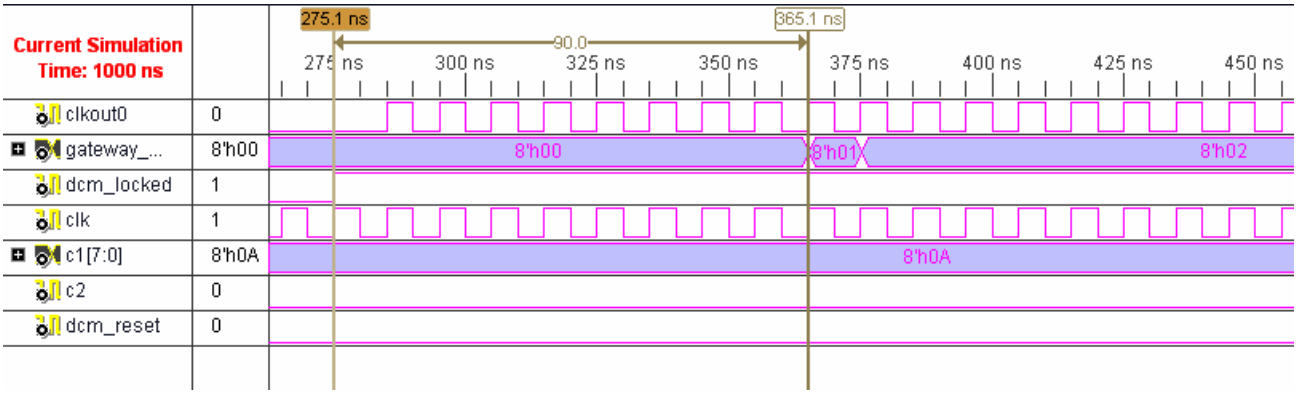


Figure (6): *Proposed structure first in first out data machine cycle*

Ser	Signal name	Signal classification	Signal Description
1	clk	**Clock** in	General System clock
2	Dcm_locked	**Control clk**	Direct clock manager to Control "output clock to the system"
3	Dcm_reset	**Control clk**	Direct clock manager reset
4	Clkout0	**Clock out**	Output clock to the system
5	C1[7:0]	**Data Input**	Serial Image Data Input
6	C2	**Data flow Control**	Selecting Image Algorithm ("0"-Gauassian , "1"-SobelX)
7	Gateway	**Data Output**	Serial Image Data Output

Table(3): *proposed FPGA Design input / output description*

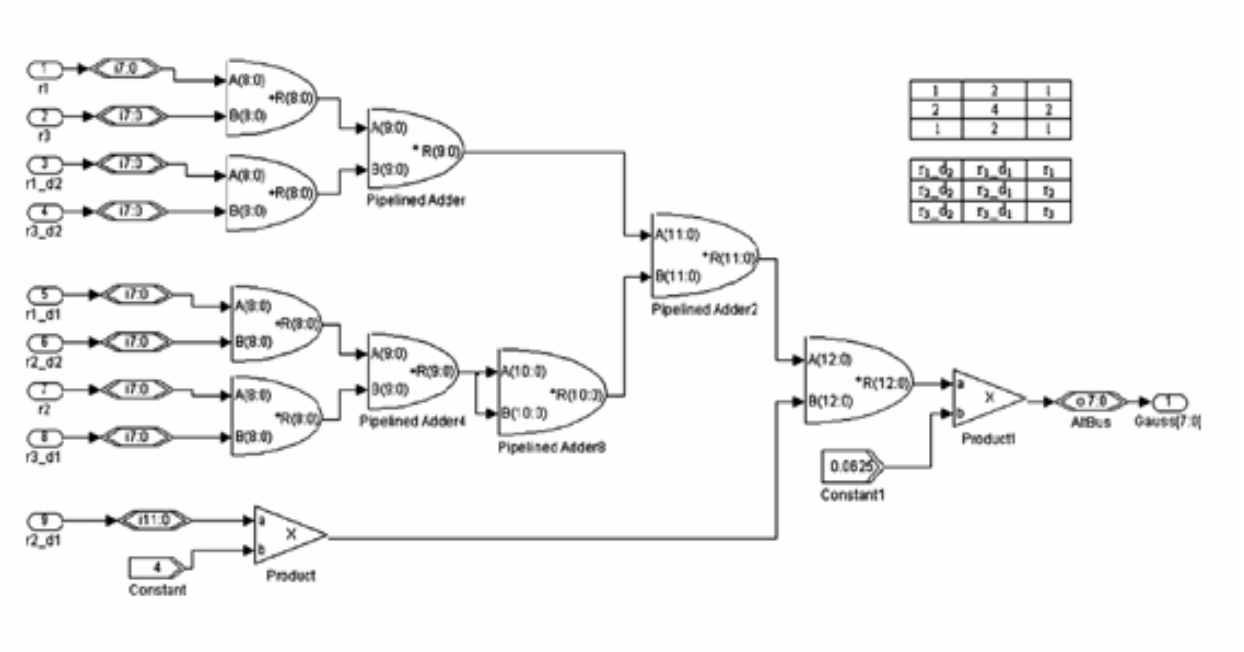
As we can see from table 2 that FPGA resources consumption due to our Design proposal for Xilinx Virtex 4 item number XC4VSX55-12FF1148 from DSP48 slices is only 2%. Finally The main target is to see the both low level vision algorithms deals with real image with the size of 640 x480 and validate the Image result and to see how accurate it can be from image vision point of view. so we verify our design by applying it into the Original image in figure 7-a and the output image result from both algorithms will be consequently appears also in figure 7-b, 7-c

a-Original image b- Gaussian Filter c- Sobel x edge detector

Figure (7): *Input/output images from the proposed general structure design Gaussian/Sobelx*

5. Comparison with other systems

In other design as shown in the following figure it used a different architecture to achieve same Gaussian filter with; less bit size (from 8bit to maximum 12bit which increases +1 bit as calculation level increases) in our design (18 in multiply 48 bit in addition at all calculation levels). For single image calculation time, it takes 3.1 mSec for calculating one image frame with size 640x480 and in our design was 3.07 mSec for the same image frame size, in the other system the hardware skeleton made especially for Gaussian filter and can not be used for other algorithms, but in our design it will be used for Gaussian and Sobelx



Figure(8): *output images from the Proposed structure Gaussian / Sobelx*

Ser	Point of comparison	Other design	Our design	Notes
1	Delay line	The same	The same	
2	Electronic elements Calculation levels	7 levels	5 levels	
3	Electronic element used for calculations	Adder , multipliers	DSP slice	
4	Maximum Number of bits used in calculations	12 bit	18 bit in multiplier and 48 bit in addition	
5	The ability to be used for other algorithm	Made especially for Gaussian algorithm need to have another schematic for other algorithm	Made for Gaussian and sobelx and can be reconfigure with slight differences for other algorithms	
6	Processing Time per frame	3.1msec	3.07msec	

Table(4): *comparison with other related system*

6. Conclusions:

The great advantages in our design is that; no need for wasting redesign time, just select from the pre-implemented equations flow by simply selecting "0" or "1" and the final output will be calculated accordingly, also more similar way of calculated algorithms can be added with slight different changes according to its kernel differences and selecting bit number can be increased accordingly, taking into consideration compact level of calculations and higher number of calculation bits.

We applicate our design in Xilinx FPGA virtex4 at frequency 100MHz in MATLAB mathworks and validate and simulate our design for the two low level algorithms Gaussian filter and Sobelx edge detector. We also convert it to Hardware Descriptive Language (HDL), Verilog and repeat checking and validation again by simulate it using Integrated Software Environment (ISE 10.1) FPGA software, as a final result shown in table 4 that speed and performance is improved and gives higher and more accurate data results.

References:

[1] D. V. Rao, S. Patil, N.A. Babu, V. Muthukumar, *Implementation and evaluation of image processing algorithms on reconfigurable architecture using C-based Hardware Descriptive Languages*, International Journal of Theoretical and Applied Computer Sciences, Vol. 1 No. 1, P .9-34, 2006

[2] R.J. Petersen, B.L. Hutchings, *An assessment of the suitability of FPGA-based systems for use in digital signal processing*, 5th International Workshop on Field-Programmable Logic and Applications, Oxford, England, P. 293-302, August 1995.

[3] B.A. Draper, J.R. Beveridge, A.P.W. Bohm, Ch. Ross, M. Chawath, *Accelerated image processing on FPGAs*, IEEE Transactions on Image Processing , December 2003.

[4] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*, Second Edition. Pearson Education International, 2002.

[5] M. Leeser, S. Miller, H. Yu, *Smart camera based on reconfigurable hardware enables diverse real time applications*, 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM2004), Napa, CA, USA, P. 147-155, 2004.

[6] I.S. Uzun, A. Amira, A. Bouridane, *FPGA implementations of fast fourier transforms for real time signal and image processing*, IEEE Proceedings Vision, Image and Signal Processing P. 283-296, 2005.

[7] N. Shirazi, P.M. Athanas, A.L. Abbot, *Implementation of a 2D fast fourier transform on a FPGA-based custom computing machine*, 5th International Workshop on Field-Programmable Logic and Applications, Vol. 975 of Lecture Notes in Computer Science, Oxford, UK, P. 282-292, August-September 1995.

[8] M. Rogers, M. Won, A. Soohoo, *Altera FPGA co-processors accelerate the performance of 3-D stereo image processing,* Altera Corporation, News & Views Spring/Summer 2005.

[9] D. Baumgartner, P. Rossler, W. Kubinger, *Perferomance Benchmark of DSP and FPGA Implementations of low level vision algorithms,* IEEE, 2007

[10] I. S. Koc. *Design considerations for real-time systems with dsp and risc architectures*. 13th European Signal Processing Conference, 2005.

[11] W.J. MacLean, *An evaluation of the suitability of FPGAs for embedded vision systems*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 3, San Diego, California, USA, June, P. 131, 2005.

[12] *Xilinx System generator for DSP Getting started guide,* release 10.1, March 2008

[13] *Xilinx ISE Help Overviw* , release 10.1, 2008.

[14] *Xilinx web site http://www.xilinx.com*

[15] *Xilinx System generator for DSP user guide,* release 10.1, March 2008

[16] *Xilinx System generator Reference manual,* release 10.1, March 2008

[17] *Xilinx ISE 10.1 in depth tutorial,* 2007.

[18] J.A. Kalomiros, J. Lygouras, *Design and evaluation of a hardware / software FPGA based system for fast image processing*, Microprocessors and Microsystems, P. 95-106, 2008.