IMPLEMENTATION OF TIME EFFICIENT SYSTEM FOR MEDIAN FILTER USING NIOS II PROCESSOR

Tanushree Selokar[1]and Narendra G. Bawane[2]
[1, 2]Department of Electronics Engineering, R.T.M.N. University, Nagpur, India

ABSTRACT
This paper is intended to develop an operative part of filtering to reduce time using Altera DE2 Cyclone II board with Altera Nios II processor. 8-bit grayscale bitmaps can be chosen to be used in image processing and make a time efficient processing. JTAG is used for serial communication to the Nios II processor for processing, and receives the processed results from Nios II processor to display the results. The algorithm is used to develop in VHDL and c code to implement it in the project. In this project NIOS II processor is selected as soft core processor which is commercial processor and Median Filter Algorithm is taken under the case study. Algorithm implemented by two different ways: software and co-design. NIOS II system is generated using SOPC builder. Programming has been done in C and NIOS II IDE is used to integrate the system. Algorithm is implemented separately using Custom Hardware to improve the performance on CYCLONE II FPGA. We are using VHDL code of the algorithm and processing it in Nios II processor to get a time efficient system.

KEYWORDS: *Co-Design, image processing, Custom Hardware, NIOS II processor.*

I. INTRODUCTION

In last few decades, embedded systems have experienced an accelerating growth both in computing power and scope of their possible applications. Moreover the designing procedure for embedded system also changed immensely. As the application demands goes on increasing with the time the complexity of the embedded system is waxing. Combination of software and hardware in design leads to improve the system performance such approach is known as Co-Design.

1.1. Co-design

Hardware/software co-design is the main technique used in the thesis. It can be defined as the cooperative design of hardware and software. The aim of co-design is to abridge the time-to-market while reducing the design effort and costs of the designed products [1].
Co-design improves the performance of the system. We are going to generate system, for this purpose we have taken an algorithm to make the performance enhance system [2,3]. Programming will be done in C and NIOS II IDE will be used to integrate the system. The soft core processor used is NIOS II.
We will be using co-design methodology as we are in need of software and hardware description of an embedded system. “Co-Design is a design methodology supporting the concurrent development of hardware and software in order to achieve system functionality and performance goals. In particular, Co-Design often refers to design activities prior to the partitioning into Hardware and Software and the activity of design partitioning itself”[4]. Figure1 shows general flow of the co-design which includes the software and hardware part which are being designed simultaneously and being worked simultaneously as they are being needed in the project to get worked.
The design flow of the general co-design approach is depicted in figure 1
Step1: The co-design process starts with specifying the system behaviour at the system level.
Step 2: After this, a pure software system will be developed to verify all algorithms.
Step 3: Performance analysis will be performed to find out the system bottlenecks.

Step 4: The hardware/software partitioning phase a plan will be made to determine which parts will realized by hardware and which parts will be realized by software. Obviously, some system bottlenecks will be replaced by hardware to improve the performance.
Step 5: Based on the results of step 4, hardware and software parts will be designed respectively.
Step 6: Co-simulation. At this step, the completed hardware and software parts will be integrated together and performance analysis will be performed.
Step 7: If the performance meets the requirements, the design can stop and if the Performance can't meet the requirements, new HW/SW partitioning and a new design.

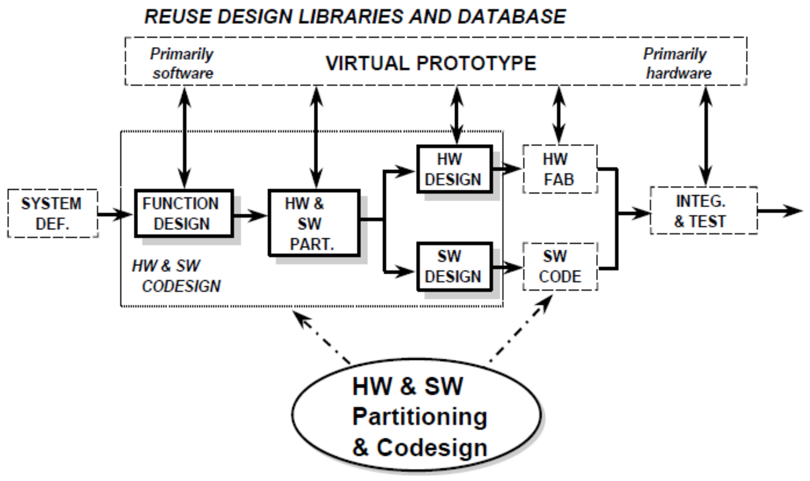


Figure 1. Co-design flow[4]

1.2.Motivation

In order to enhance the performance of the system, Co-design method is frequently used. In process of making an easier explore design tradeoffs, continual verification throughout the design cycle, mutual influence of both HW and SW early in the design cycle, partitioning (Co-design) is the only way to get all this features[4].The advanced ASIC and FPGA technologies facilitate the integration of complex Systems on a Chip (SoC). The use of different Co-design methods is application specific. In our project work we required an opaque and fast system[1]. The Nios II soft-core processor have the highest operating frequency it also allows reuse of code and highly configurable. These esteem features motivate implementation using Co-design with NIOS II soft core processor and study the performance parameters.

1.3.Aim And Objective

The objective of our project is

1. To design the median filter algorithm in VHDL and C.
2. To design the system for implementing of the algorithm.
3. Partitioning using co-design methodology by custom hardware to get the time efficient system.

II. METHODOLOGY: MEDIAN FILTER ALGORITHM

Median filter is a spatial filtering operation, so it uses a 2-D mask that is applied to each pixel in the input image. It is used to remove defects and noise from pictures. Median filter is much less sensitive than the mean to extreme values (called outliers), therefore it is better without reducing the sharpness of the image and edge preserving nature. To apply the mask means to centre it in a pixel, evaluating the covered pixel brightness and determining which brightness value is the median value. The algorithm is: every pixel from the picture to be filtered is replaced by the median value of the neighbouring pixels[1]. The picture is thus transformed by the median filter by another picture that has exactly the same size. For every pixel P of the input picture we first create a list of the 9 (3x3)

pixels surrounding P. The 9 pixels are then sorted. The median value is the value located at the center of the sorted list. The pixel P in the filtered picture takes this median value. In our example the pictures are grayscale pictures, 8 bits per pixel. The pixel values are between 0 (black) and 255 (white). In the last step, this module will be integrated into FPGA boards together a Nios II CPU.
The median value is determined by placing the brightness in ascending order and selecting the center value. The obtained median value will be the value for that pixel in the output image. The oldest sample is discarded, a new sample acquired, and the calculation repeats.

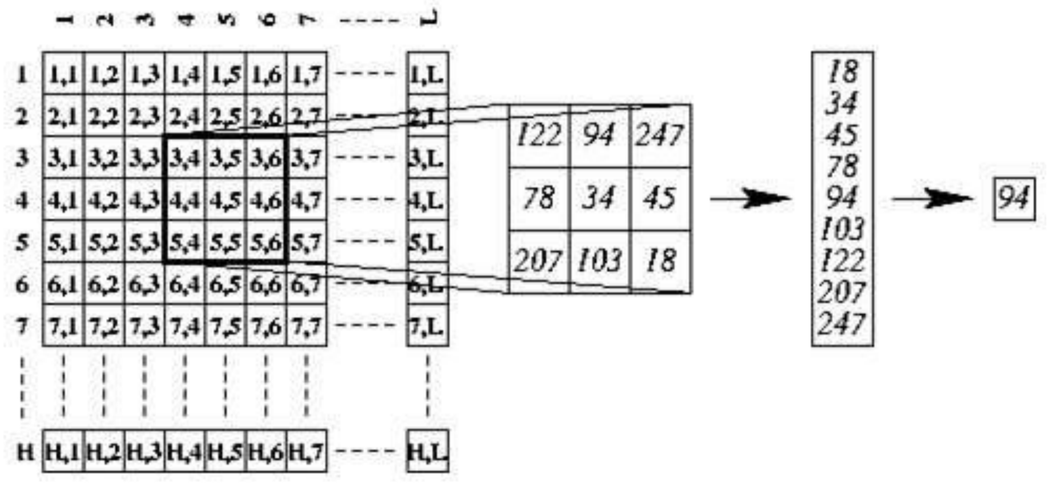


Figure 2. An example of the median filter application, as in this case, habitually a 3x3 median filter is used[9].

Let [X, Y] be the pixel located at row X, column Y of a picture. The value of pixel [4, 5] in the filtered picture is computed from the 3x3 neighbourhood of the pixel [4, 5] of the source picture, that is: [3]
The list of pixels: {[3,4],[3,5],[3,6],[4,4],[4,5],[4,6],[5,4],[5,5],[5,6]}.
The list of the pixel values: {122, 94, 247, 78, 34, 45, 207, 103, 18}.
The sorted list is {18, 34, 45, 78, 94, 103, 122, 207, 247}.
The median value is 94 so the value of the pixel in the filtered picture will be 94.

III. RESULTS: IMPLEMENTATION OF ALGORITHM ON FPGA USING NIOS II PROCESSOR

Algorithm will be implemented on CYCLON II FPGA using NIOS II Processor and the software used are Quartus II and NIOS II IDE.
NIOS II is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture. The processor is highly flexible in any design configuration, and mainly suitable for system-on-a-chip (SOC) designs. The NIOS II IDE have GNU compiler with C/C++ license, Following are some features of NIOS II Soft core Processor.

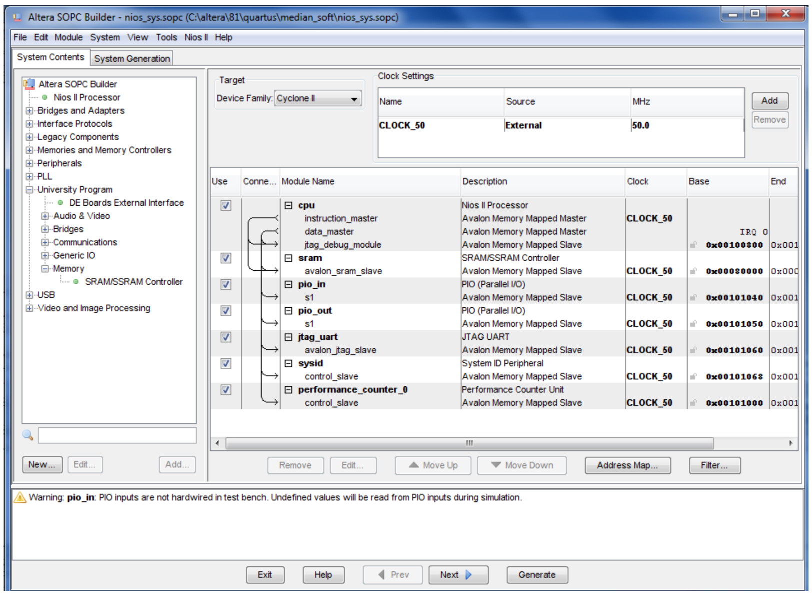
- Soft IP Core : A soft-core processor is a microprocessor fully described in software, usually in an HDL, which can be synthesized in programmable hardware, such as FPGAs.
- Reduced Instruction Set Computer (RISC)8/16/32-bits memory controller for external PROM and SRAM.
- No pipeline, 5 or 6 stages pipeline configurations.
- It has full 32-bit instruction set.
- 32 general-purpose registers.
- For more interrupt sources it has external interrupt controller interface.
- Single-instruction 32 × 32 multiply and divide producing a 32-bit result.
- For computing 64-bit and 128-bit products dedicated instructions of multiplication is given.
- Floating-point Custom instructions for single-precision floating-point operations.

3.1. Design Steps For Implementation

The system components required for Software/Hardware implementation of Median filter algorithm on FPGA in QUARTUS II software.

- NIOS II Processor
- JTAG UART
- SRAM MEMORY
- Input ,Output GPIO'S
- Performance Counter

SOPC Builder automatically generates the interconnect logic to integrate the components in the hardware system. It can be selected from a list of standard processor cores and components provided with the Nios II EDS [5]. Following window shows the selection of components required and its system generation.



Target
Device Family: Cyclone II

Clock Settings

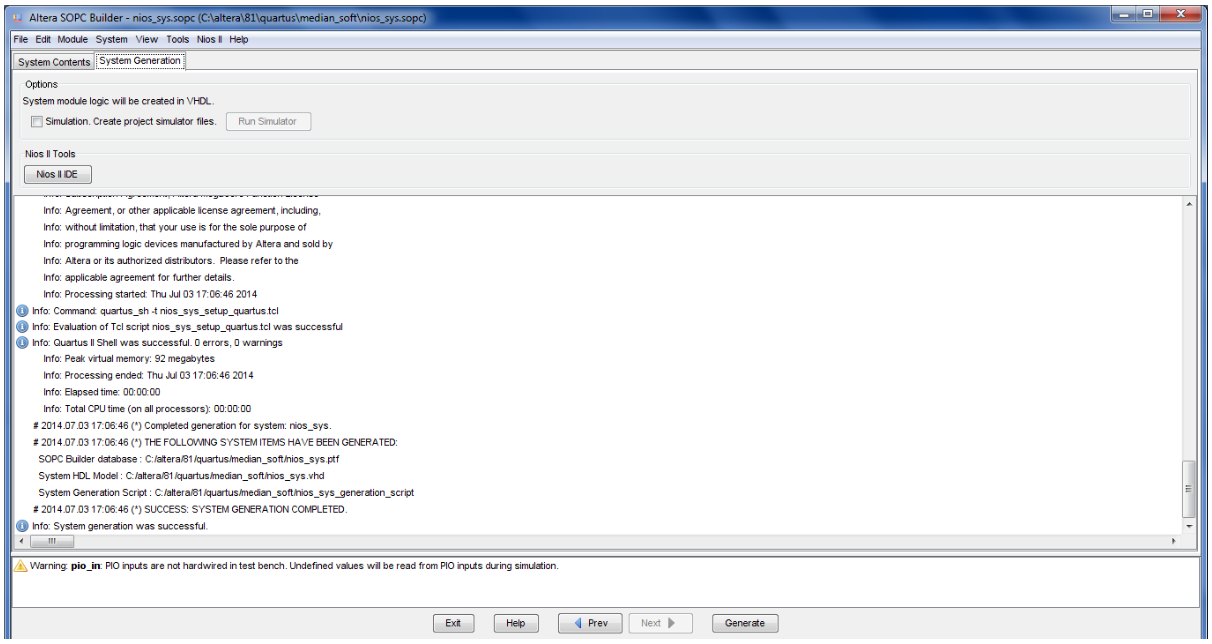
Name	Source	MHz
CLOCK_50	External	50.0

Use	Conne...	Module Name	Description	Clock	Base	End
✓		cpu	Nios II Processor			
		instruction_master	Avalon Memory Mapped Master	CLOCK_50		
		data_master	Avalon Memory Mapped Master		IRQ 0	
		jtag_debug_module	Avalon Memory Mapped Slave		0x00100800	0x001
✓		sram	SRAM/SSRAM Controller			
		avalon_sram_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00080000	0x000
✓		pio_in	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00101040	0x001
✓		pio_out	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00101050	0x001
✓		jtag_uart	JTAG UART			
		avalon_jtag_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00101060	0x001
✓		sysid	System ID Peripheral			
		control_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00101068	0x001
✓		performance_counter_0	Performance Counter Unit			
		control_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00101000	0x001

Warning: **pio_in**: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

Figure 3. System contents in SOPC Builder

In the Figure 3, the components required to build a system are taken in SOPC Builder.


Altera SOPC Builder - nios_sys.sopc (C:\altera\81\quartus\median_soft\nios_sys.sopc)
File Edit Module System View Tools Nios II Help
System Contents
System Generation
Run Simulator
Nios II Tools
Nios II IDE
Exit
Help
Prev
Next
Generate

Options
System module logic will be created in VHDL.
☐ Simulation. Create project simulator files.

```
    Info: Agreement, or other applicable license agreement, including,
    Info: without limitation, that your use is for the sole purpose of
    Info: programming logic devices manufactured by Altera and sold by
    Info: Altera or its authorized distributors.  Please refer to the
    Info: applicable agreement for further details.
    Info: Processing started: Thu Jul 03 17:06:46 2014
Info: Command: quartus_sh -t nios_sys_setup_quartus.tcl
Info: Evaluation of Tcl script nios_sys_setup_quartus.tcl was successful
Info: Quartus II Shell was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 92 megabytes
    Info: Processing ended: Thu Jul 03 17:06:46 2014
    Info: Elapsed time: 00:00:00
    Info: Total CPU time (on all processors): 00:00:00
  # 2014.07.03 17:06:46 (*) Completed generation for system: nios_sys.
  # 2014.07.03 17:06:46 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED:
   SOPC Builder database : C:/altera/81/quartus/median_soft/nios_sys.ptf
   System HDL Model : C:/altera/81/quartus/median_soft/nios_sys.vhd
   System Generation Script : C:/altera/81/quartus/median_soft/nios_sys_generation_script
  # 2014.07.03 17:06:46 (*) SUCCESS: SYSTEM GENERATION COMPLETED.
Info: System generation was successful.
```

Warning: **pio_in**: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

Figure 4. System Generation without custom hardware

In Figure 4, the successful generation of the system is shown.

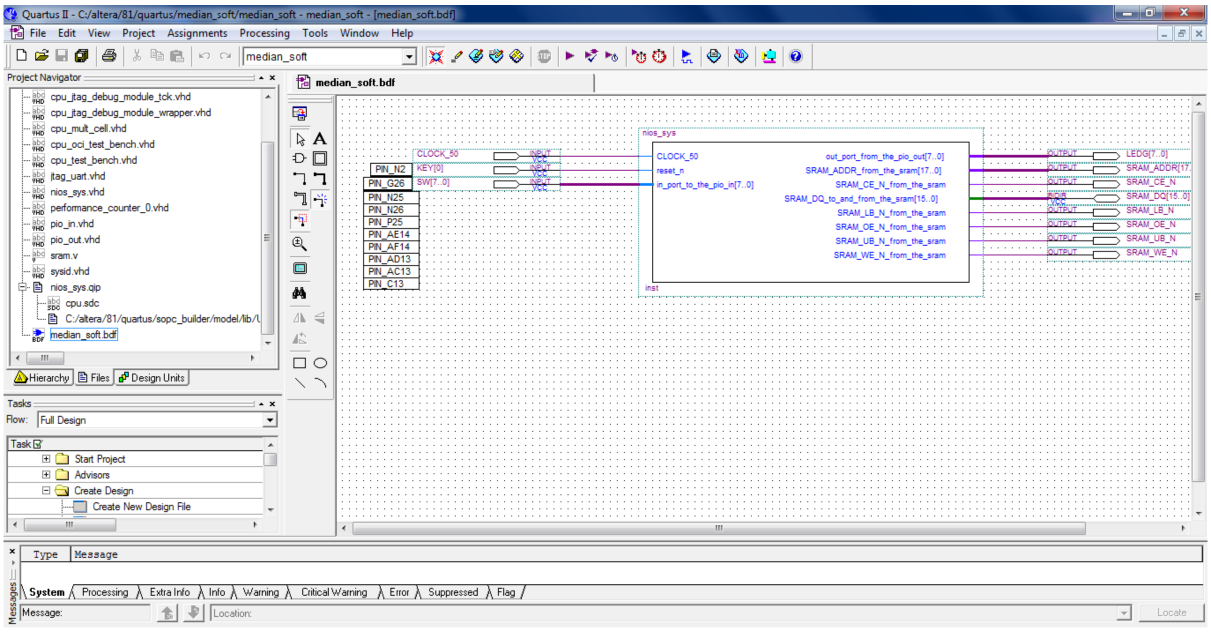


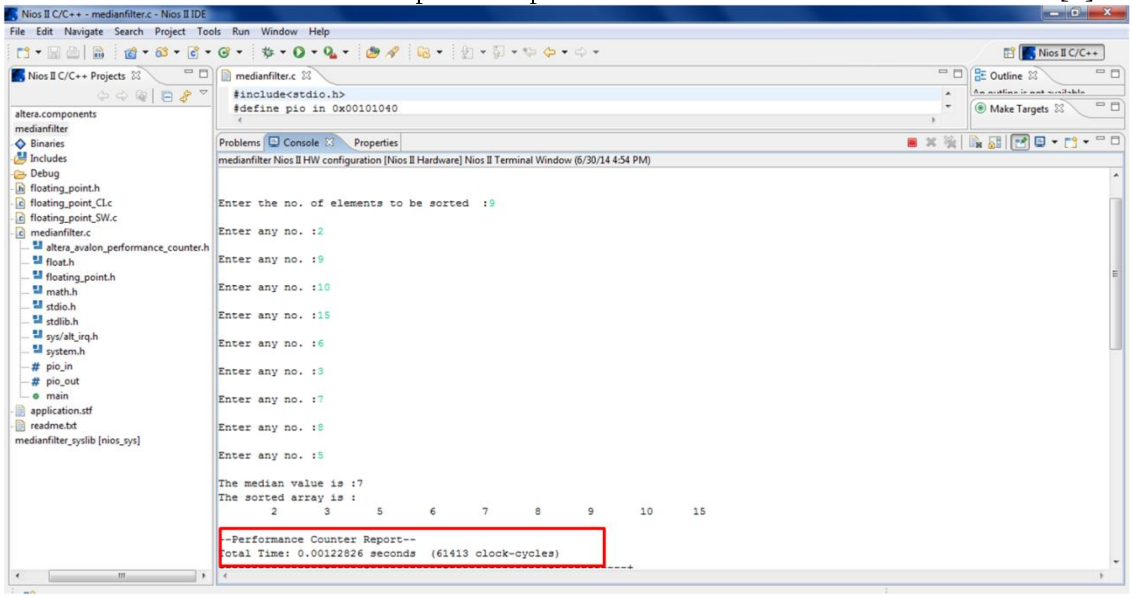
Figure 5. NIOS System without custom hardware Block Diagram File view.

In Figure 5, integration of SOPC Builder with Quartus Software is shown. In this BDF the pin assignment is done by importing the pin assignment of Cyclone II (EP2C35F672C6) FPGA.

The generation of system in hardware is complete. Here the connection between Cyclone II FPGA and host computer is done by cable USB-Blaster. After successful hardware generation time limited file is generated.

3.2. Software flow using NIOS II System

The results for median filter using NIOS II IDE in c-code has given below. After Generation of system in Quartus II software the **.sopcinfo** file is called up in NIOS II IDE as a hardware platform. The new project is made in NIOS II IDE, then the project for C-code is chosen in which the "hello_world" template is selected. The C-code for Median filter Algorithm is written in **Hello_world.c** file. Then the project is build using Build Project command. After the project build the code will be implemented on CYCLONE II (EP2C35F672C6) FPGA using command **Run as NIOS II Hardware**. After all these steps the output will be seen in NIOS II Console Window[6].



```
#include<stdio.h>
#define pio in 0x00101040
```

```
medianfilter Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (6/30/14 4:54 PM)

Enter the no. of elements to be sorted  :9

Enter any no. :2

Enter any no. :9

Enter any no. :10

Enter any no. :15

Enter any no. :6

Enter any no. :3

Enter any no. :7

Enter any no. :8

Enter any no. :5

The median value is :7
The sorted array is :
        2       3       5       6       7       8       9       10      15

--Performance Counter Report--
Total Time: 0.00122826 seconds  (61413 clock-cycles)
```

Figure 6. Nios II Processor Software Flow

In Figure 6, the output of implementation of Algorithm on FPGA is shown with clock cycles and time required for execution of Algorithm. The table no.1 shows the clock cycles required for software flow.

Table 1. Clock Cycles for execution of Algorithm in software

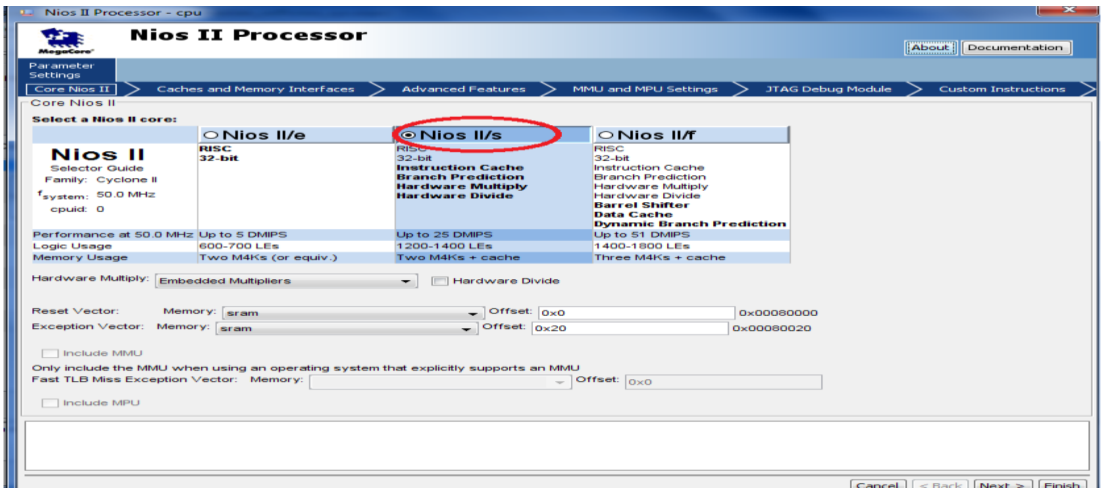
Median filter Algorithm	Hardware required in percentage	Clock Cycles	Time Required
Software	3110/33216 (9%)	61413clk cycles	0.00122826 sec

IV. Results: Implementation Of Algorithm With Custom Hardware On Fpga Using Nios II Processor

In hardware implementation, interfacing of median filter and Nios II processor is done. While designing a system that includesaNios II embedded processor, we can accelerate time-critical software algorithms by interfacing custom hardware to the Nios II processor.

4.1. Design Steps For The Hardware Implementation

After software implementation same procedure has to be followed with some changes in system.
1. For hardware implementation system is generated in SOPC builder.
2. Implementation of Algorithm is done on FPGA using NIOS II IDE.
3. Inputs are being provided from the FPGA board and we run it in NIOS II IDE.
4. The program calculates the processing time and throughput for each of the versions, to demonstrate the improved efficiency of a custom hardware compared to software implementation.



Core Nios II

Select a Nios II core:

Nios II Selector Guide Family: Cyclone II f_{system}: 50.0 MHz cpuid: 0	Nios II/e	[✓ Nios II/s]	Nios II/f
	RISC 32-bit	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Prediction
Performance at 50.0 MHz	Up to 5 DMIPS	Up to 25 DMIPS	Up to 51 DMIPS
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs
Memory Usage	Two M4Ks (or equiv.)	Two M4Ks + cache	Three M4Ks + cache

Hardware Multiply: Embedded Multipliers ☐ Hardware Divide

Reset Vector: Memory: sram Offset: 0x0 0x00080000
Exception Vector: Memory: sram Offset: 0x20 0x00080020

☐ Include MMU
Only include the MMU when using an operating system that explicitly supports an MMU
Fast TLB Miss Exception Vector: Memory: Offset: 0x0

☐ Include MPU

Figure 7. Nios II Processor is selected as before for Custom Hardware Flow.

Figure 7 shows selection of in NIOS II Processor. The inclusion of floating point hardware in processor keeping all other peripherals same, it leads to increase in hardware. It maps the memory location from SRAM interface in SOPC builder. For data transfer it uses the 32-bit internal registers if Nios II processor[7].

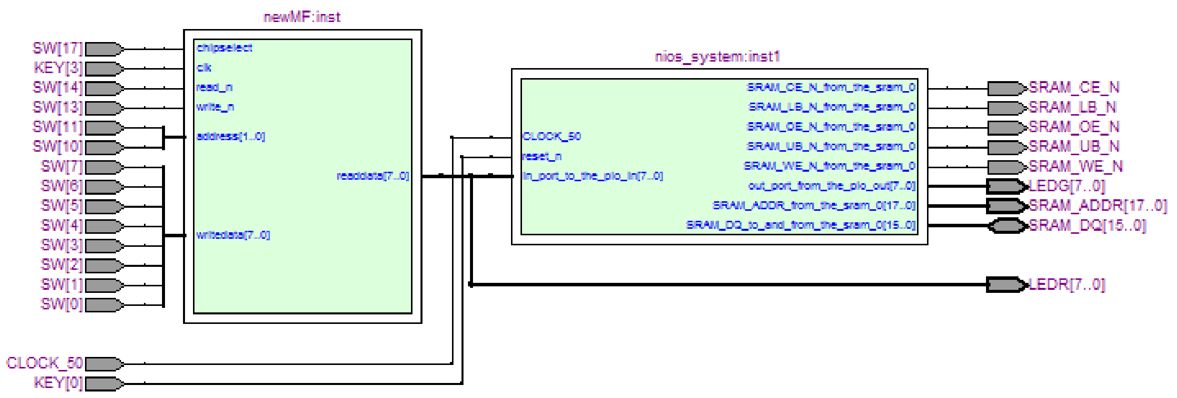
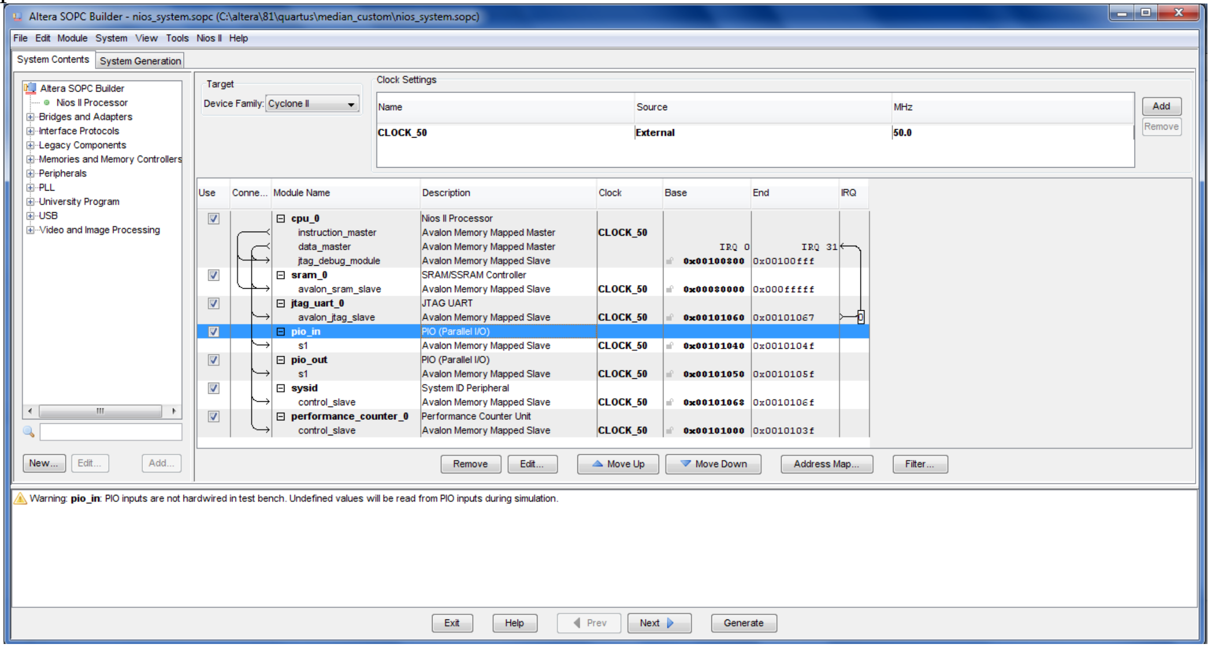


Figure 8. RTL view of Custom Hardware

Figure 8 shows the RTL of Custom Hardware. The Nios II embedded Design Suite (EDS) generates a macro in the system header file, system.h. You can use the macro directly in your C or C++ application code, and you do not need to program assembly code to access custom hardware. Software can also invoke custom instructions in Nios II processor assembly language. Custom Hardware is added as floating point hardware in NIOS II processor. It contains following files[8].
floating_point.c— main program
floating_point.h—global definitions
floating_point_CI.c—functions to exercise the floating-point custom instructions
floating_point_SW.c—functions to exercise the software-implemented floating-point operations.

4.2. Results Of Median Filter Algorithm Using Custom Hardware

The System is generated in SOPC Builder. In this system the custom hardware is added as floating point hardware.


Altera SOPC Builder - nios_system.sopc (C:\altera\81\quartus\median_custom\nios_system.sopc)

File Edit Module System View Tools Nios II Help

System Contents | System Generation

Altera SOPC Builder
- Nios II Processor
- Bridges and Adapters
- Interface Protocols
- Legacy Components
- Memories and Memory Controllers
- Peripherals
- PLL
- University Program
- USB
- Video and Image Processing

Target
Device Family: Cyclone II

Clock Settings

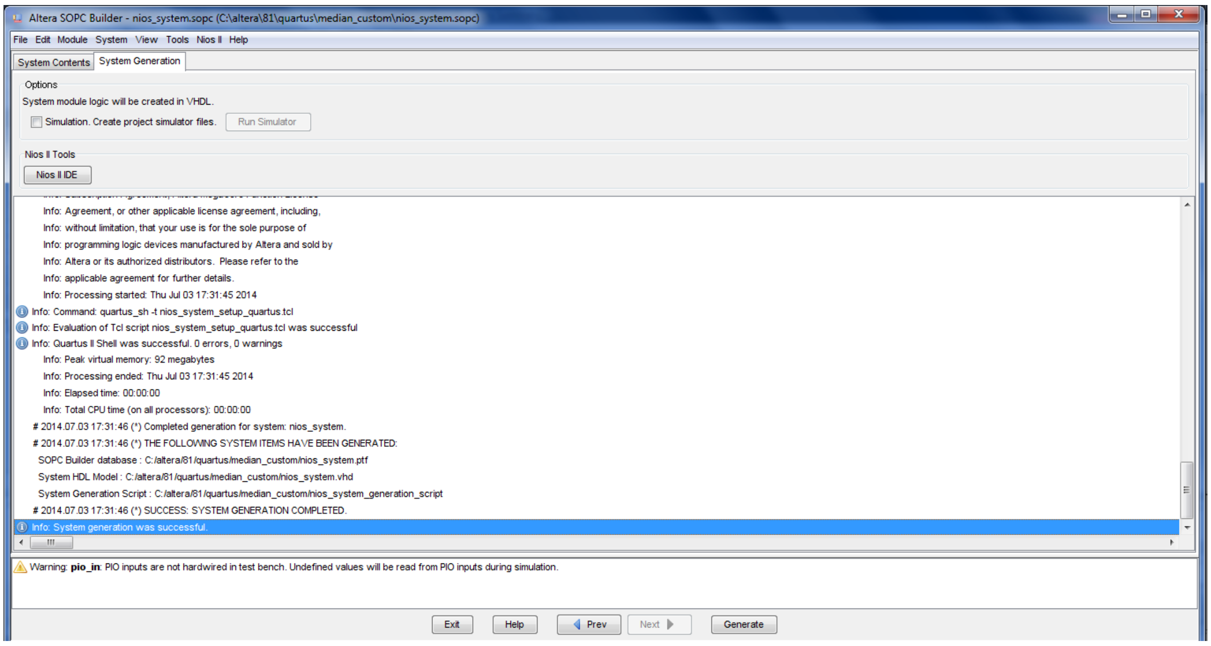
Name	Source	MHz
CLOCK_50	External	50.0

Use	Conne...	Module Name	Description	Clock	Base	End	IRQ
✓		cpu_0	Nios II Processor				
		instruction_master	Avalon Memory Mapped Master	CLOCK_50			
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Memory Mapped Slave		0x00100800	0x00100fff	
✓		sram_0	SRAM/SSRAM Controller				
		avalon_sram_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00080000	0x000fffff	
✓		jtag_uart_0	JTAG UART				
		avalon_jtag_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00101060	0x00101067	0
✓		pio_in	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00101040	0x0010104f	
✓		pio_out	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00101050	0x0010105f	
✓		sysid	System ID Peripheral				
		control_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00101068	0x0010106f	
✓		performance_counter_0	Performance Counter Unit				
		control_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00101000	0x0010103f	

Warning: **pio_in**: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

New... Edit... Add...
Add Remove
Remove Edit... Move Up Move Down Address Map... Filter...
Exit Help Prev Next Generate

Figure 9. System for custom hardware flow in SOPC Builder



Altera SOPC Builder - nios_system.sopc (C:\altera\81\quartus\median_custom\nios_system.sopc)
File Edit Module System View Tools Nios II Help
System Contents
System Generation
Options
Run Simulator
Nios II Tools
Nios II IDE
Exit
Help
Prev
Next
Generate

System module logic will be created in VHDL.

Simulation. Create project simulator files.

```
    Info: Agreement, or other applicable license agreement, including,
    Info: without limitation, that your use is for the sole purpose of
    Info: programming logic devices manufactured by Altera and sold by
    Info: Altera or its authorized distributors.  Please refer to the
    Info: applicable agreement for further details.
    Info: Processing started: Thu Jul 03 17:31:45 2014
Info: Command: quartus_sh -t nios_system_setup_quartus.tcl
Info: Evaluation of Tcl script nios_system_setup_quartus.tcl was successful
Info: Quartus II Shell was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 92 megabytes
    Info: Processing ended: Thu Jul 03 17:31:45 2014
    Info: Elapsed time: 00:00:00
    Info: Total CPU time (on all processors): 00:00:00
  # 2014.07.03 17:31:46 (*) Completed generation for system: nios_system.
  # 2014.07.03 17:31:46 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED:
   SOPC Builder database : C:/altera/81/quartus/median_custom/nios_system.ptf
   System HDL Model : C:/altera/81/quartus/median_custom/nios_system.vhd
   System Generation Script : C:/altera/81/quartus/median_custom/nios_system_generation_script
  # 2014.07.03 17:31:46 (*) SUCCESS: SYSTEM GENERATION COMPLETED.
Info: System generation was successful.

Warning: pio_in: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
```

Figure 10. System Generation for Custom Hardware

Figure 9 and Figure 10 shows the system contents and generation of system in SOPC Builder. After generation of system in SOPC, the pin assignment and compilation is done in QUARTUS II.

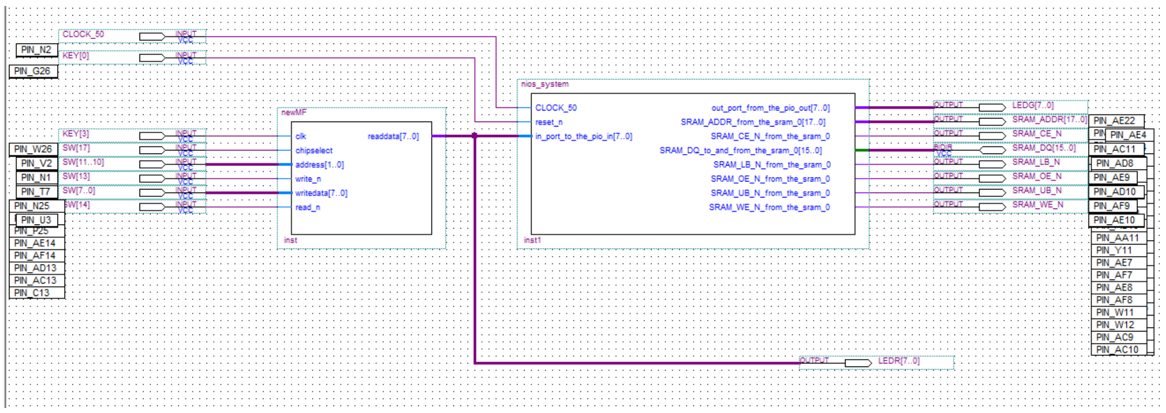
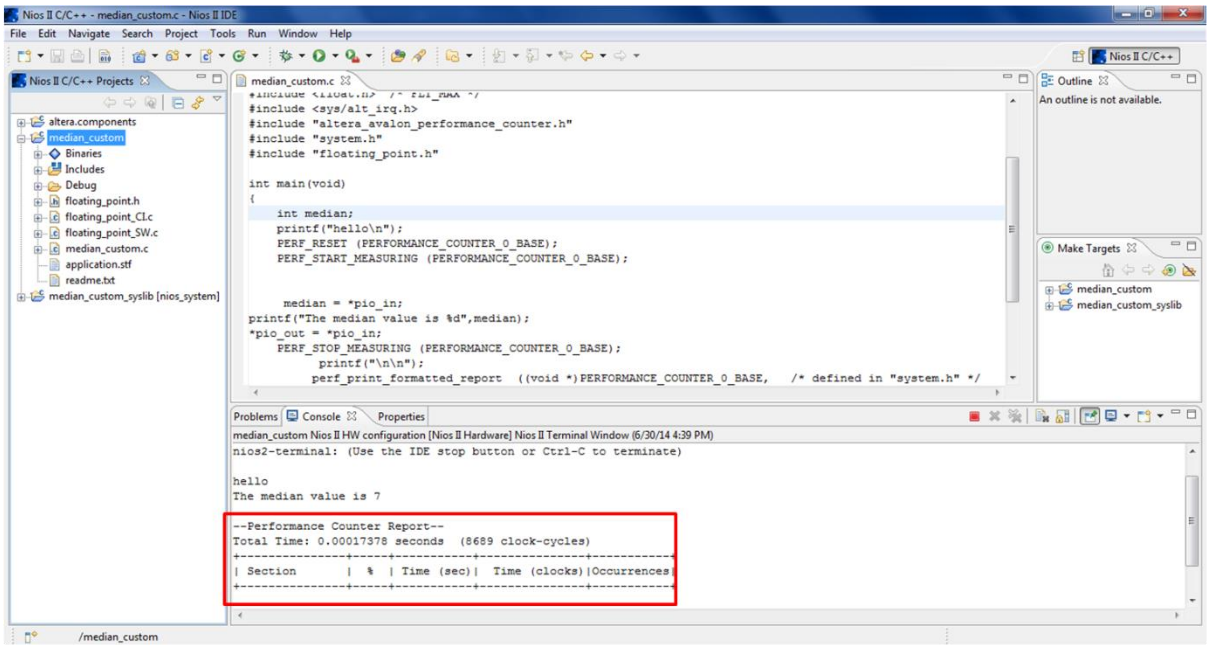


Figure 11. NIOS System with custom hardware Block Diagram File view

In Quartus II, The pin assignment is done by importing the file of CYCLONE II (EP2C35F672C6) FPGA. In the figure 11 the address lines from SRAM memory are assigned to custom hardware. After the successful compilation of system, hardware generation in CYCLONE II FPGA is done and time limited file is generated. After hardware generation of system using CYCLONE II FPGA, the algorithm is implemented in NIOS II IDE. We have to follow same steps as stated with addition of Custom Hardware files in NIOS II IDE Project.
After generating hello_world.cfile, we have to add custom instruction files floating_point.c, floating_point.h, floating_point_CI.c and floating_point_SW.cin the project. Then the project is build by command Build Project. While building a project, a macro function system.h is generated which connects the C-code of algorithm to Custom hardware. After successful build of project the Algorithm is implemented on CYCLONE II (EP2C35F672C6) FPGA. This implementation is done by command Run as NIOS II Hardware. After execution of this command the result is shown in NIOS II console window.



```
#include <float.h>  /* FLT_MAX */
#include <sys/alt_irq.h>
#include "altera_avalon_performance_counter.h"
#include "system.h"
#include "floating_point.h"

int main(void)
{
    int median;
    printf("hello\n");
    PERF_RESET (PERFORMANCE_COUNTER_0_BASE);
    PERF_START_MEASURING (PERFORMANCE_COUNTER_0_BASE);


     median = *pio_in;
printf("The median value is %d",median);
*pio_out = *pio_in;
    PERF_STOP_MEASURING (PERFORMANCE_COUNTER_0_BASE);
          printf("\n\n");
        perf_print_formatted_report  ((void *)PERFORMANCE_COUNTER_0_BASE,   /* defined in "system.h" */
```

```
median_custom Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (6/30/14 4:39 PM)
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

hello
The median value is 7

--Performance Counter Report--
Total Time: 0.00017378 seconds  (8689 clock-cycles)
+---------------+-----+-----------+---------------+-----------
| Section       |  %  | Time (sec)|  Time (clocks)|Occurrences|
+---------------+-----+-----------+---------------+-----------
```

Figure 12. Median Filter result with Custom Hardware

In Figure 12, the output of implementation of Algorithm on FPGA is shown with clock cycles and time required for execution of Algorithm. The table 2 shows the clock cycles required with custom hardware.

Table 2. Clock Cycles for execution of Algorithm in Hardware

Algorithm	Hardware required in Percentage	Clock Cycles	Time Required
Custom Hardware	3238/33216 (10%)	8689 clk cycles	0.00017378 sec

V. CONCLUSION

Median Filter Algorithm considered as a case study is implemented using the hardware / software co-design methodology. Hardware / software co-design methodology implementation gives an optimized design of the algorithm. Algorithm is implemented on CYCLONE II FPGA based around NIOS II processor.
85% efficiency is achieved by using custom hardware. Also, when Algorithm is implemented with NIOS II system using Custom Hardware, speed of the Median filter algorithm is increased but area required for the implementation is also increased.

5.1. Synthesis Report

After execution of Algorithm on FPGA with and without custom hardware, their comparison is done for parameters clock cycles. From the table no. 1 and table no. 2 the comparison is done shown in table3.

Table 3. CPU Clock cycles and time required

Items	**Clock Cycles**	**Time Required**
Algorithm Without Custom Hardware	61413	0.00122826 Sec
Algorithm With Custom Hardware	8689	0.00017378 Sec

From the performance analysis results of clock cycles and time required for execution in software is more as compared with execution of algorithm in hardware. The system which is generated using SOPC Builder is compiled in Quartus II software. The hardware required for generation of system is depending upon the LE's used in CYCLONE II (EP2C35F672C6) FPGA. The comparison in hardware change is shown in table 4.

Table 4. Comparison of compilation report

Items	Total Count	Without Custom Hardware	With Custom Hardware
Total Logic Elements	33216	3110(9%)	3238(10%)
Total Combinational Functions	33216	2934(9%)	3044(9%)
Dedicated Logic Registers	33216	1915(6%)	2018(6%)
Total Pins	475	61(13%)	75(16%)
Total Memory Bits	483840	46208(10%)	46208(10%)
Embedded multipliers 9-bit Elements	70	4(6%)	4(6%)

The above table shows the comparison between the software and hardware systems i.e. system with and without custom hardware. It shows that inclusion of custom hardware increases the hardware which gives better result in terms of clock cycles and require for execution of algorithm.

VI. FUTURE SCOPE

As the selected Processor is soft core processor, enabling change of hardware according to the application. Optimizations in area, required for Median filter Algorithm; can be obtained by designing various optimization approaches for the various blocks of the algorithm. Here, Median filter algorithm is accelerated using custom hardware with NIOS II processor, in future acceleration image processing can be done by median filter. It can also work with the custom instruction.

REFERENCES

[1] G. Bosman "A Survey of Co-Design Ideas and Methodologies", Vriji University, Amsterdam, July, 2005.
[2] "Nios II Processor Handbook", Altera Corporation, 2007.
[3] Jason G. Tong, Ian D. L. Anderson and Mohammed A. S. Khalid, "Soft-Core Processors for Embedded Systems" University of Windsor, The 18th International Conference on Microelectronics (ICM) 2006.
[4] "Rolf Ernst "Co design of Embedded Systems: Status and Trends", Braunschweig University of Technology, IEEE Design & Test of computers, April–June 1998
[5] Ery Arias-Castro and David L. Donoho, "Does Median Filtering truly preserve edges better than linear filtering" University of California, San Diego and Stanford University, The Annals of Statistics,2009, Vol. 37, No. 3, 1172–1206.
[6] Altera Corporation, "NIOS II Software Developer's Handbook",2011.
[7] Nallaperumal.K,Varghese.J, Saudia.S , Annam.S, Kumar.P, "Iterative Adaptive Switching Median Filter", Industrial Electronics and Applications, Singapore, 1ST IEEE Conference onAltera Corporation, "NIOS II Processor Reference Handbook,"2010.
[8] AnisBoudabous, Ahmed Ben Atitallah, LazharKhriji, Patrice Kadionik, and NouriMasmoudi, "HW/SWDesign-Based Implementation of Vector Median Rational Hybrid Filter", Sultan Qaboos University, Oman, University Bordeaux I, France, The International Arab Journal of Information Technology, Vol. 7, No. 1, January 2010.
[9] Chen Kah Yee, "Median Filter Using NIOS II Processor With Sort Hardware Accelerator", University Technology Malaysia, May 2009.

AUTHOR BIOGRAPHY

Tanushree Selokar Date of Birth: 27-11-1989Education: M.Tech. in Electronics from Nagpur University, Nagpur India B.E in Electronics from Nagpur University, Umrer, India

Narendra G Bawane, M. Tech. (IIT, Delhi), Ph. D. (VNIT)
He has total teaching experience of more than 23 years at graduate and Post-graduate level. His areas of interest are Artificial Neural Network (ANN), Embedded system, Fuzzy logic system, Wavelet analysis, Hybrid intelligence, Image processing & Emotion in speech and facial recognition, Biomedical engineering etc.