Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 24 (2016) 1127 – 1134

Procedia Technology

International Conference on Emerging Trends in Engineering, Science and Technology (ICETEST - 2015)

Parallel Computation Using DSP Slices in FPGA

Supriya Unnikrishnan K[a] , Sudheesh Madhavan[b]

[a]*PG Scholar, ECE Dept., Toc H Institute Of Science and Technology , Kochi,India*
[b]*Director ,Tecnode Solutions Pvt Ltd, Bangalore,India*

Abstract

The sophistication of applications and hunger for high quality digital data demands increase in the processing power. High performance signal processing is possible only though parallelism. At the same time, flexibility and scalability are the need of the hour due to dynamically changing standards and design up gradation. This paper describes an implementation of the computational framework using the DSP Slices in the FPGA. The customized instructions will provide the computation flexibility whereas specialized DSP macros in FPGA ensure high performance.

Peer-review under responsibility of the organizing committee of ICETEST – 2015

Keywords: FPGA , DSP , Parallel Processing

1. PARALLEL COMPUTATION

An array of tightly coupled processing elements connected in a specified topology like mesh, delta, etc. with a common system clock scales up processing speed in DSP algorithms which has inherent parallelism [1,2]. The granularity of processing element, memory management, interconnection topology, etc. is key design areas of such array processors [3, 4]. Out of the four processing modes classified by Flynn namely Single Instruction Single Data (SISD), Single Instruction Multiple Data (SIMD), Multiple Instruction Single Data (MISD) and Multiple Instruction Multiple Data (MIMD), SIMD and MIMD are most promising for Digital Signal Processing (DSP) applications [5]. SIMD is useful for applications processing large arrays and matrices. MIMD is used when pipelined processing needs to be done on data like DCT, quantization and Huffman Encoding etc. as in JPEG algorithm.

Over the last few decades, good amount of research work is done on the different array processor architectures targeting parallel computation [6, 7, 8, 9]. Most of them use a customized processing element, connected in a systematic fashion with provision for data storage and synchronized control. With Field Programmable Gate Arrays (FPGAs) entering the hardware acceleration arena, reconfigurable parallel computation is focused on. FPGA vendors provide generic processing elements for speeding up the development process. This paper describes a parallel computation processor using the generic DSP processing element embedded in FPGAs.

2. DSP SLICE BASED SYSTEM FOR PARALLEL COMPUTATION

DSP Slices are high performance computation macros available in most of the leading FPGAs [9,10,11].In this work, DSP slice enabled parallel computation is implemented using Virtex5LX50T FPGA. Virtex5LXT have 46 DSP Slices available as columns embedded in FPGA. A wrap module is made over the DSP Slice; these slices are connected together in configurable fashion and are controlled using instructions which are a superset of DSP Slice instructions. Currently, many platforms like Simulink are available where we can drag and drop the DSP blocks and connect them as per the algorithm. But this flow lacks the controllability and visibility of the user into the design. When we stitch up the DSP slices as an array processor, the

* Corresponding author. *E-mail address:*ksupriya_unni@yahoomail.com

Peer-review under responsibility of the organizing committee of ICETEST – 2015
doi:10.1016/j.protcy.2016.05.064

controllability and configurability is increased. The number of rows and columns of the array and the mode of operation can be configured at RTL level. Currently SIMD and MIMD mode is supported.

3. USER SCENARIOS

The scenarios in which we can use the DSP Slice processor are given below

- *Online Data Processing :* The data from array sensors is directly latched in the DSP Slice registers. The computation is performed as per the program stored in instruction memory and processed data are passed on at the output interface. This simple scheme can be used when the rate at which data is receiving is sufficiently low.

- *Microblaze As Controller :*A full-fledged system can be built over DSP Slice array processor using the soft processor like Microblaze available in FPGA. The program/data transfer, control and synchronization of array processor can be done by Microblaze. Multiport DDR interfaces can be used for external storage. The interface between Microblaze and DSP array processor is set of host interface registers.

- *Coprocessor :* FPGA based card with backplane interfaces like PCI can be used as coprocessor card where the DSP Slice array acts as a coprocessor sharing memory with the CPU. This accelerates the computation intensive tasks of CPU.

4. DESIGN

The architecture of array processor is shown in the figure 1. The arrays are arranged in 2D mesh with 4 nearest neighbor connection. Sixteen bit operations are done, though DSP can support up to 32 bits. The operations are controlled by a finite state machine (FSM).Block Random Access Memory (BRAM) available in FPGA is used for storing operand1, operand2, instruction and output . The code is written in Verilog. The IPs (Intellectual Property) for DSP48E and BRAM are generated using the Xilinx Core generator. The number of rows and columns in an array and the number of array processor instances (MIMD mode) in a design can be configured using Perl at the RTL level. This is done using defines generated by Perl program. The arguments from command line is used to generate define file that is included in the RTL data base. ROWSIZE, COLSIZE and mode of operation are modified as per this include file.

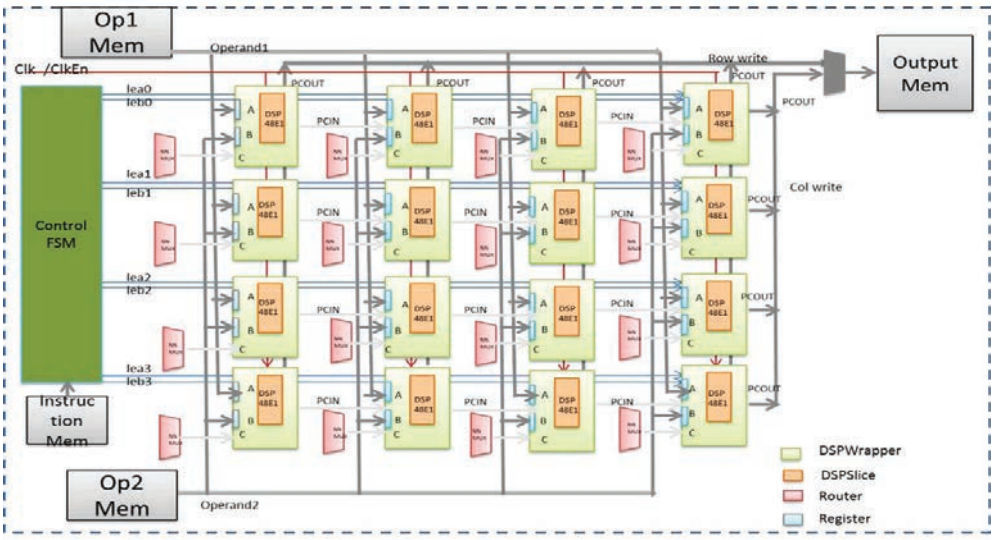

Fig. 1. DSP Slice Array Processor Architecture

4.1. DSP Slice

The math operation of the DSP Slice consists of 25 bit by 18 bit two's complement multiplier followed by three 48 bit three path multiplexers [11,12] .This is followed by three input adder/subtract or two input logic unit. DSP Slice Configuration is through Core Generator, an integrated tool of Xilinx ISE.DSP Slice can be configured for 64 unique instructions selected from a list of available instructions. The simplified diagram of DSP Slice from the user guide [11] is shown in figure 2.

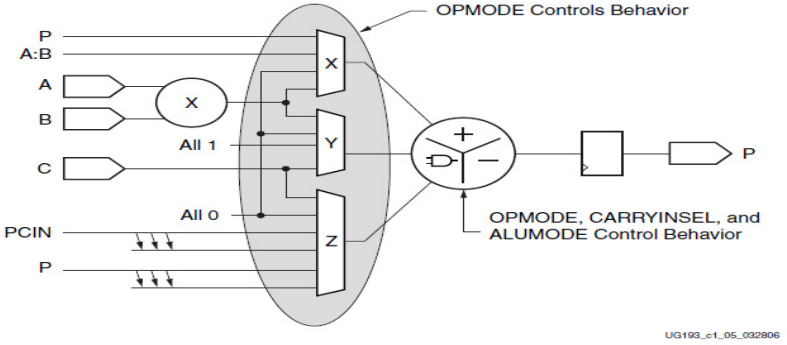

Fig. 2. DSP Slice Diagram [11]

4.2. Instruction Encoding

A 128 bit instruction word is defined for the microprogramming of the DSP Slice array. Instruction word is formatted in such a way that DSP Slice array can be operated in SIMD as well as MIMD mode ; i.e. one instruction for each DSP Slice. Fig 3 provides encoding details. In addition to DSP48E instructions, customized instructions for memory access, control and data movement are defined.

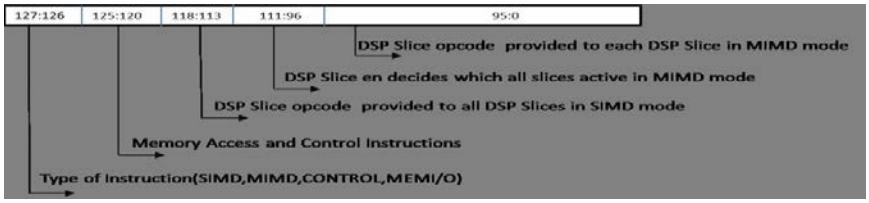

Fig. 3. DSP Slice Array Processor Architecture

4.3. Finite State Machine

The finite state machine acts as the synchronizer between all DSP Slices. It also sequences the actions.FSM does the instruction read, decode and generation of control signals for the DSP array, Memory and Router.

4.4. Memory

Memory is arranged as stitch up of small blocks of 256x16 bit BRAMs. These are arranged as rows and columns with a common controller that controls row to row, column to row, row to column and column to column read/write. Only sequential memory access is supported. The instances of memory increase with configuration i.e. number of array rows and columns.This is a power aware mapping where only the required memory blocks can be activated and remaining memory blocks are gated thus saving power. For example, if we had used a 1024x16 memory block in the first column instead of 4 256x1 blocks, on a row read all the four 1024x16 memory blocks need to be activated. Another reason for providing small memory units is to be able to read a row or a column or data for a 4x4 array in a single read. This minimizes the bottleneck introduced by memory bandwidth. The data to be used by all 16 slices is available in the registers associated with memory (shown in Figure 4).

4.5. DSP Wrapper

This is the wrap around module with DSP slice instance and registers for holding operands and output. This is added as part of modularization so that the interface to the processing element can be generic and independent of the FPGA specific DSP Slice interface. There is provision for bypassing registers.

4.6. Router

This is an intelligent mux that selects the data from Upper, Lower, Right and Left nearest neighbor .These data movements are required in many DSP algorithms. The selection to the mux is generated by FSM after instruction decoding .The routing to nearest left neighbor is embedded in FPGA architecture. The signal PCOUT of DSP Slice is cascaded to PCIN of the DSP Slice to the right. Hence, while selecting algorithm to implement, algorithm with more horizontal movements is more optimal. The other connections are added in the design by the user.

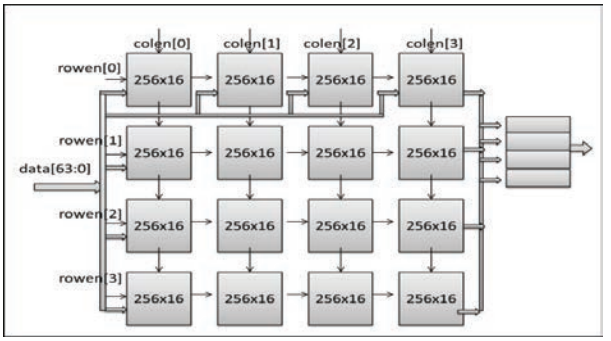

Fig 4. Memory Organization

5. MATRIX MULTIPLICATION

The calculations used for benchmarking includes matrix multiplication and Digital Filtering. Figure illustrates the steps in a 4x4 matrix multiplication.

a. First row of the matrix is loaded into 4x4 Slice array register A of all processing elements. The 64 bit data from Op1 memory is broadcasted to all four rows of DSP Slices. The latch enable (le) for all rows are asserted thereby enabling data to be stored in the register.

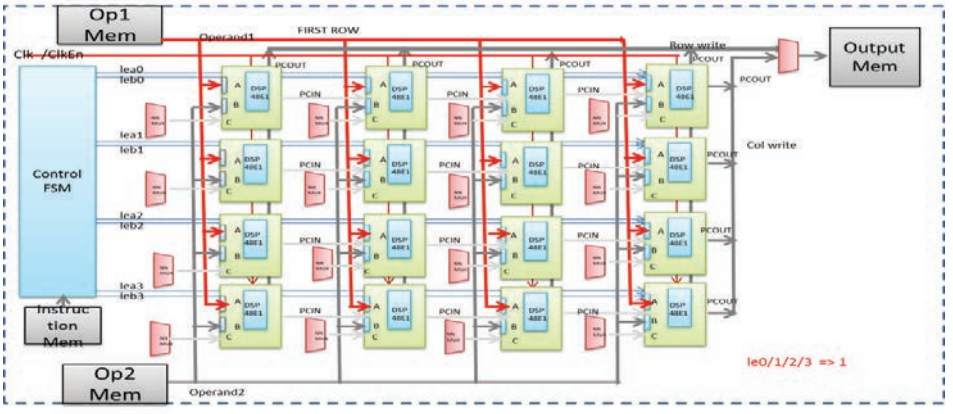

Fig. 5. First Matrix Row loaded in all slice registers

b. Four columns of matrix are latched in to the B register of the rows of array processor. For this, Op2 memory is read four times.

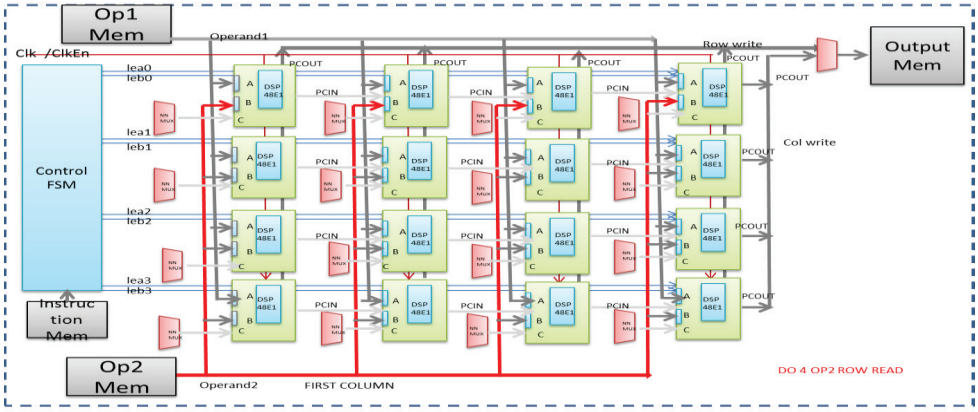

Fig. 6. Matrix column loaded in to Slice registers

c. Execute PCIN +A*B instruction in all DSP Slices which gives the sum of partial products. The first row elements of product matrix are available at the PCOUT of the last column of processing elements.

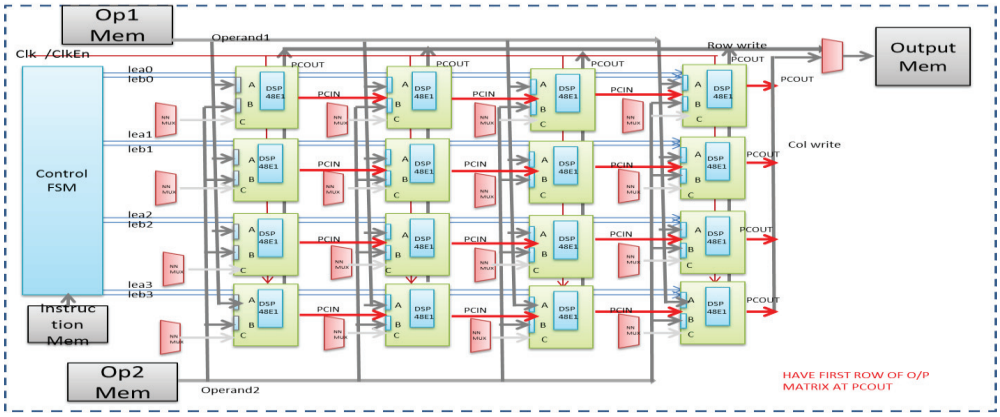

Fig. 7. Sum of Partial Products

d. The output of each row is written into memory as a row. The above steps need to be repeated four times to get four rows of the product matrix.

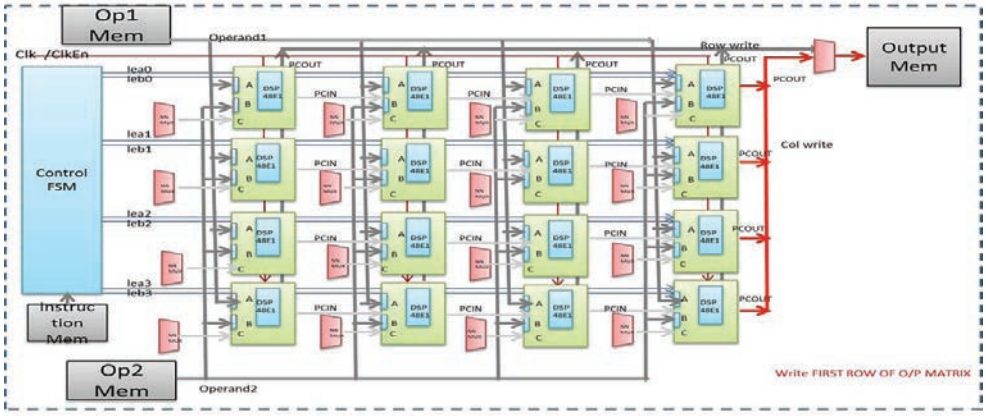

Fig 8. Output Matrix row written to output memory

5.1. Simulation

The microprogram for matrix multiplication is loaded into instruction memory and operands are loaded into operand memory from text files using *$readmemh* task in Verilog. Test bench is created and simulated in ISIM (Xilinx Simulation Tool).The same matrix data are given to Matlab program which does the matrix multiplication. The verification suite is as shown in the figure 9. Both behavioral and Post Route simulation with standard delay format (sdf) annotation is done before testing on the board.

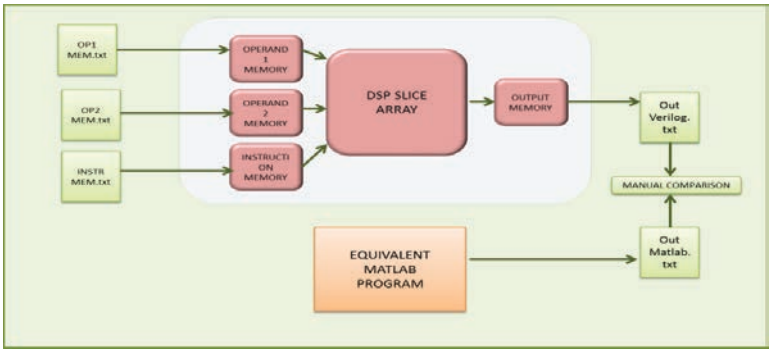

Fig. 9. Verification Strategy

6. DIGITAL FILTER

Four tap Fixed point filter is designed using *fir1* function in Matlab. A square wave is given as input. The same coefficients and input are given to the DSP Slice Array processor. The output from Matlab and DSP is plotted. The direct form FIR filter output samples are obtained from the equation given below.

$$Y(k) = a(1)*x(k) + a(2)*x(k-1) + a(3)*x(k-2) + a(4)*x(k-3) \quad (1)$$

The data samples and coefficients are stored as a column in the Op1 and Op2 memory. A column to row read of coefficients is done and broadcasted to all four rows of slice array. The column to row read of data sample is followed by an address increment. By doing this four times, we get the time shifted samples loaded in all the four rows of slice array. Once data population is done, the sum of partial products gives the output value [9].Fixed-point signed number is used with word length 16 and fraction length 15. The output value written into output memory is captured to text file. This text file is given to the Matlab program for plotting. The input, Matlab output and DSP array processor output are plotted for comparison.

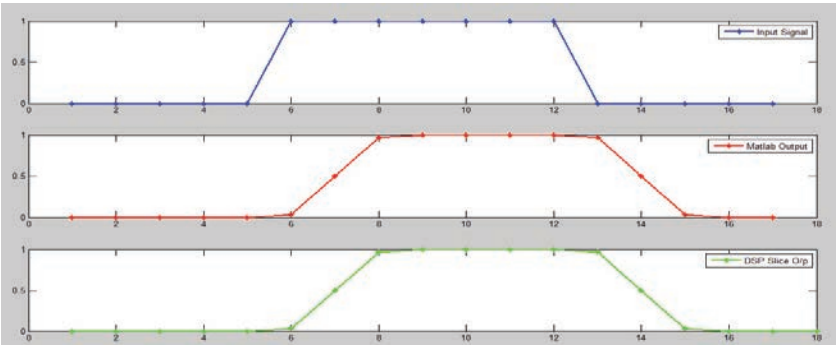

Fig. 10 Input and Output Plots

7. FPGA IMPLEMENTATION

The Implementation is done on Virtex5 FPGA ML505 Evaluation board which has 46 DSP Slices arranged in columns. These Slices in a column are interconnected internally using cascaded inputs without using FPGA fabric. The matrix and program are initialized in the memory using Xilinx .coe file. The timing constraint is given in the user constraint file (ucf) .The location constraint for DSP Slices need to be given so that Slices in the same row are interconnected using internal fabric. After implementation; the design is analyzed for power, timing and utilization.

7.1. POWER

Since power is one of the major concerns in FPGA based designs, Power analysis is done using Xilinx XPower Analyzer .The netlist, constraints and switching activity files are the input to Xpower. The Switching Activity Interchange Format (SAIF) file should be generated with test case which gives the maximum toggle in the internal signals.In the ISIM simulation window run a Post route simulation. Once ISIM console pops up give commands as shown in the snapshot below .The switching activity details are recorded in default file xpower.saif.

```
ISim> saif open -allnets
ISim> run 1us
ISim> saif close
```

Once SAIF file is generated, go to Xilinx ISE -> Tools -> XPower Analyzer ->Open Design .Specify input files Native Circuit Description(NCD), Physical Constraints File(PCF) and Switching Activity Interchange Format(SAIF) file here and run a power analysis. The output report window is shown in Figure 11.This is the power calculated for a 4x4 matrix multiplication program. As can be seen, Clock and BRAMS and leakage power are the major factors consuming power .

File Edit View Tools Help

Device	
Family	Virtex5
Part	xc5vlx50t
Package	ff1136
Grade	Commercial
Process	Typical
Speed Grade	-2

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.099	1	—	—
Logic	0.002	1923	28800	7
Signals	0.003	7084	—	—
BRAMs	0.044	12	60	20
DSPs	0.001	22	48	46
IOs	0.001	6	480	1
Leakage	0.557			
Total	0.707			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.7	83.8	51.2

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	0.551	0.149	0.402
Vccaux	2.500	0.062	0.000	0.062
Vcco25	2.500	0.000	0.000	0.000

Supply Power (W)	Total	Dynamic	Quiescent
	0.707	0.150	0.557

Fig. 11 Power Analysis Result from Xpower

Different strategies that affect power consumption are analyzed. If the application doesn't use multiplications we can use USE_MULT attribute by which multiplier is not enabled and hence power is reduced. Using cascade paths inside DSP Slice instead of fabric routing is another method to reduce power. Different cases are analyzed for power consumption (figure 12).

Case1: This is the highest consumption case in which Enable pins of BRAMs are not used and clock enables to DSP Slices are not used.

Case 2: Enables to BRAMs are used and BRAMs are enabled only during reads and writes.

Case3: Enables to BRAMs and clock gating to DSP Slices are used

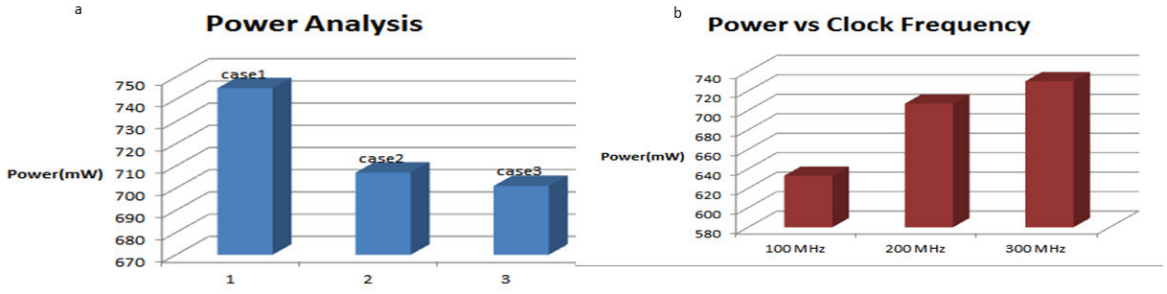

Fig. 12 a) Power Variation with different power saving methods b) Power Variation with Clock Frequency

Design is analyzed for different clock frequencies in case3 configuration. As expected, power increases with clock frequency and it is for the designer to decide on power and frequency tradeoff for the application.

7.2. TIMING AND UTILIZATION

The timing and utilization report is shown in figure 1. The data port registers and pipeling in DSP Slice allows users to tradeoff between latency and increased clock frequency. This design doesn't use internal port registers and achieved frequency is 287 MHz.

There are 46 DSP Slices available in virtex5LXT out of which 22 are used. This report is for 4x4 array processor and hence 16 slices are used in the array and rest 6 are used for the computations used in the logic used .

a

```
All constraints were met.

Data Sheet report:
------------------
All values displayed in nanoseconds (ns)

Clock to Setup on destination clock clk_i
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk_i          |   3.484|         |         |         |
---------------+---------+---------+---------+---------+

Timing summary:
---------------

Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)

Constraints cover 12850 paths, 0 nets, and 8441 connections

Design statistics:
   Minimum period:   3.484ns{1}   (Maximum frequency: 287.026MHz)
```

b

Utilization Summary

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	3,672	28,800	12%
Number used as Flip Flops	3,672		
Number of Slice LUTs	1,925	28,800	6%
Number used as logic	1,925	28,800	6%
Number using O6 output only	1,922		
Number using O5 and O6	3		
Number of occupied Slices	1,425	7,200	19%
Number of LUT Flip Flop pairs used	5,172		
Number with an unused Flip Flop	1,500	5,172	29%
Number with an unused LUT	3,247	5,172	62%
Number of fully used LUT-FF pairs	425	5,172	8%
Number of unique control sets	21		
Number of slice register sites lost to control set restrictions	20	28,800	1%
Number of bonded IOBs	6	480	1%
Number of BlockRAM/FIFO	12	60	20%
Number using BlockRAM only	12		
Number of 36k BlockRAM used	12		
Total Memory used (KB)	432	2,160	20%
Number of BUFG/BUFGCTRLs	2	32	6%
Number used as BUFGs	2		
Number of DSP48Es	22	48	45%
Average Fanout of Non-Clock Nets	2.43		

Fig. 13 a) Design Utilization Summary ISE b) Timing Report Summary

8. SCOPE FOR FUTURE WORK

The DSP Slice processor can be extended with high level custom instructions and compiler. GUI based configuration for different modes of the DSP Slice processor will ease the customization for different applications. The router can be upgraded to support different interconnect topology. FPGA based parallel computation provides a compromise between speed of application specific fixed hardware and the flexibility of programmability in Processors. When designed with FPGAs specialized for signal processing like Virtex-SX series, high end data processing like image/video processing can be done on this platform. These designs can be extended as a coprocessor platform .The power and performance comparison with existing array processor architectures is also area of research interest.

References

[1] R. Tessier, W. Burleson, "Reconfigurable Computing and Digital Signal Processing: Past, Present, and Future", Programmable Digital Signal Processors, Yu Wen Hu, ed., Marcel Dekker, pp. 147--186, 2002.

[2] Walt Kester , Mixed-Signal and DSP Design Techniques, ISBN: 978-0-7506-7611-3 , paperback edition,December 2002,pp.191-244

[3] Compton, C., Hauck, S. et Al. An Introduction to Reconfigurable Computing. IEEE Computer (April 2000).

[4] L. J. Karam , I. AlKamal , A. Gatherer , G. A. Frantz , D. V. Anderson and B. L. Evans "Trends in multicore DSP platforms", IEEE Signal Process. Mag., vol. 26, no. 6, pp.38 -49 2009

[5] A.D. Kshemkalyani, M. Singhal, Distributed Computing: Principles, Algorithms, and Systems, ISBN: 9780521189842,paper back edition, Cambridge University Press,2011,pp1-49

[6] R. Hartenstein, "A Decade of Reconfigurable Computing: A Visionary Retrospective," Proc. Design, Automation and Test in Europe (DATE 01), IEEE CS Press, 2001, pp. 642-649.

[7] Yu, Z. (2010) 'Towards High-Performance and Energy-Efficient Multi-core Processors', in K. Iniewski (ed.), CMOS Processors and Memories. Dordrecht et al.: Springer.

[8] Mahendra Pratap Singh and Manoj Kumar Jain. Article: A Survey of Reconfigurable Architectures. International Journal of Computer Applications 98(14):35-40, July 2014

[9] Supriya Unnikrishnan K, Augustine Abraham,Sudheesh Madhavan, Miny G, Jeyamma, Ralph D Kappithan ."Data Processing Platform with DSP Slice Based Processor" International Journal of Engineering Research and Technology, ISSN 2278-0181, Vol3,Issue 12,December 2014

[10] H. Y. Cheah, S. A. Fahmy, D. L. Maskell, and C. Kulkarni, "A lean FPGA soft processor built using a DSP block," in Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), 2012, pp. 237-240.

[11] Virtex5 FPGA XtremeDSP Design Considerations User Guide v3.5 ,2012.www.xilinx.com/support/documentation/user_guides/ug193.pdf

[12] Xilinx Xcell Journal Issue 59 ,2006 *Virtex5 Special Edition* .http://www.xilinx.com/publications/archives/xcell/Xcell59.pdf