ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

DESIGN AND COMPARISON OF RISC PROCESSORS USING DIFFERENT ALU ARCHITECTURES

Sunitha M S[1], Bharat G Hegde[2], Deepakakumar N Hegde[2]

Associate Professor, Dept of E.C.E, P.E.S Institute of Technology West Campus, Bangalore, India[1]

Graduate Student, Dept of E.C.E, P.E.S Institute of Technology West Campus, Bangalore, India[2]

ABSTRACT: Building low-power, high speed systems have been in demand, in recent years, because of the fast growing technologies in mobile communication and computation. Arithmetic and Logic Unit is a core component of almost all computing machines and processors. This work involves the design and comparison of 3 different 16 bit RISC processors based on 3 different ALU architectures. The ALU architectures are differentiated based on consumer requirements, keeping in mind cost, speed and power. Comparisons are done on aspects of area occupied, speed and power consumption. The first ALU design is an economical design, less complex and low power. The second design is a high speed, low power model using Carry Look-Ahead Adders (CLAs) and Vedic multiplier. Extensive parallelism is seen in them. The third design is targeted for low power and compactness and uses Prefix adders and Booth multiplier. The design of the RISC processors also involves the design of memory and development of opcodes which are also included in this work. A typical RISC based program on circular convolution has been implemented on all of these for comparison purposes. Results prove that ALU2 is the fastest and also consumes very less power. It is 15% faster than ALU1 but also occupies 20% more area. ALU3 occupies 30% lesser area than ALU1 and is a low power model. Though moderate in performance, ALU1 is the easiest to design.

Keywords: ALU, RISC Processor, Pipeline Breakage, Vedic Multiplier, Convolution.

I. INTRODUCTION

The architecture of ALU has serious impact on timing, power dissipation and area. The power dissipation of the ALU depends mainly on the architecture. The timing and area depends on the type of circuits used to implement the ALU components. The battery technology does not advance at the same rate as the microelectronics technology. So designers are faced with constraints such as high speed, high throughput, small silicon area, and at the same time, low-power consumption. Hence proper choice of ALU architecture is needed. Although many functions can be performed by ALU, the basic arithmetic operations of addition, subtraction, multiplication, division and simple Boolean operations such as logical and shifting operations continue to be the core operations. An ALU loads data from the input registers or memory. Based on the type of control signal from the control unit, it performs the desired computation on this data and finally stores the result in an output register. The control unit is responsible for moving the processed data between these registers, ALU and memory.

A RISC processor is designed to work with a small but powerful set of instructions, with the aim of increasing the throughput of the processor. The main feature of the RISC processor is its ability to support single cycle operation, meaning that the instruction is fetched from the instruction memory at the maximum speed of the memory.

The intention of this work was to create the functional units of the RISC processor as building blocks in Verilog HDL [1] since at a high level of complexity it is easier to implement the function in software. After designing different ALU structures, we combine them with memory and register set to perform as a full fledged RISC processor. The different ALU designs are validated by implementation of Circular Convolution on these. The opcodes designed for this purpose are also discussed in this paper.

The rest of the paper is organised as follows. Section II briefly describes the design of the 3 ALU architectures. Section III describes the design of the RISC processor. Convolution program implementation is described in Section IV. FPGA implementation and results are discussed in Section V. Section VI concludes the paper.

II. DESIGN OF ALU ARCHITECTURES

ALU1 (economical) is built in a simplistic way without tendering to power, delay or area constraints. It consists of a Ripple Carry Adder (16 bit) built with full adders. It's a slow adder as each stage has to wait for the previous carry to

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

occur. The multiplier used is an Array multiplier. This increases the speed in comparison to iterative addition but is still not optimal. A n x n multiplier requires n^2 AND gates, n half adders and n x(n-2) full adders.

ALU2 (high speed, low power) is built using constraints of parallelism. The Carry Look Ahead adder (CLA) used in this consists of carry generate and propagate terms that are used to precompute the carry thereby increasing its speed [2]. The hardware complexity, though, is very high as can be seen in the results. Hence the area occupied is large. The novelty involved is in the use of Vedic multiplier using *Urdhava Triyagbhyam sutra*. Further it is known that the conventional Vedic multiplication hardware, has some limitations. Hence to overcome those limitations, a novel approach has been taken with the use of unique 'addition tree' structure to add partially generated products [3]. A 2x2 multiplier is built with basic gates using this sutra as shown in Fig 1. This is then used to create a 4x4 multiplier as shown in Fig 2. This is further expanded to build a 16x16 multiplier. As will be seen in the results, the speed and area occupancy of this multiplier are appreciable making it a viable design for high speed digital signal processing applications.

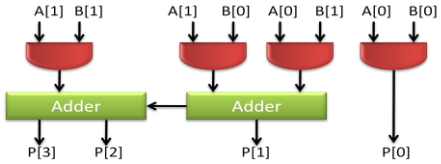


Fig 1. A 2x2 Vedic multiplier

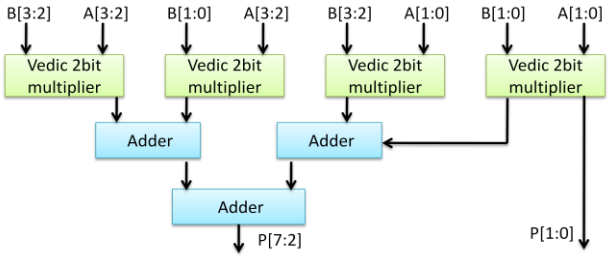


Fig. 2 A 4-bit Vedic multiplier using 2-bit multipliers

ALU3 (low power, compact) is designed with prefix adders considered to be the most optimum amongst high speed low power adders. It uses J. Sklansky's prefix adder structure as shown in Fig 3. The delay is just $\log_2 N$ [4]. Hence it is faster and also less complex in hardware in comparison to CLA's.

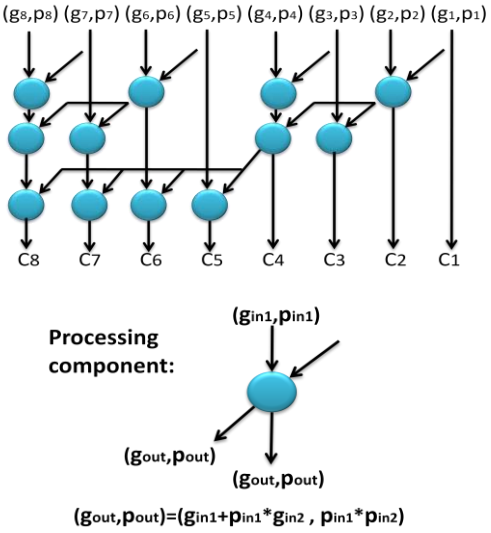


Fig. 3 Sklansky's Prefix Graph

Multiplication is implemented using the Booth multiplier shown in Fig 4[5]. This is slow compared to other multipliers and a sequential design in itself. Since 16 clock edges are required for one multiplication operation to complete, a

ISSN: 2319-8753
International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

different clock in verilog module is used which is 20 times faster than the system clock. This is accomplished using #time delay statements. Since it is based on iterative algorithm, very less hardware is used. Practically, a PLL can be used for clocking.

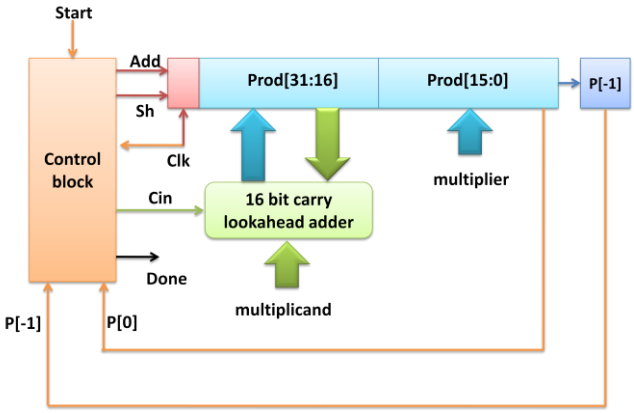


Fig. 4 Booth Multiplier

The Divider design is based on iterative subtraction (shift &subtract). The block diagram of such a divider is shown in Fig 5. The divider used in this work is a 16-bit by 8-bit divider [6].

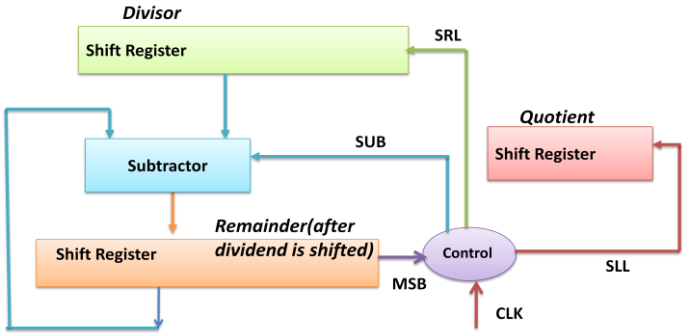


Fig. 5 Block diagram of divider

Shift and rotate operations are taken care of by Barrel shifter built using 2:1multiplexer as shown in Fig 6. This is one of the fastest and the simplest way of doing it.

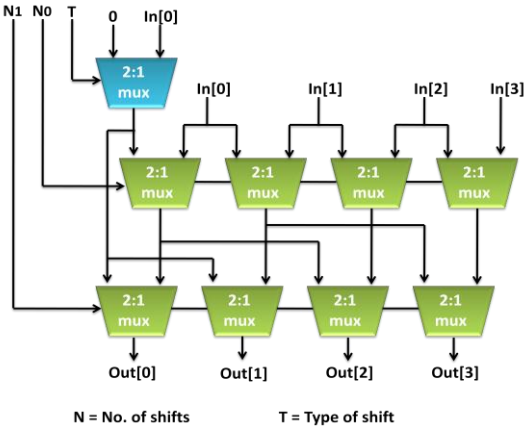


Fig. 6 A 4-bit barrel shifter

Logical unit is not taken as a separate module but statements and built-in functions of verilog are made use of. Altogether 8 logical operations can be done, with bit-reset being a special one. In order to facilitate full-fledged processor operation, memory is a must. A dual port SRAM is modelled in verilog to hold data for processing and a

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

ROM like memory is also inculcated for storing the instruction set. The opcodes are stored in these and fetched during operation.
The status/flag register is a hardware register which contains information about the state of the processor. This design makes use of carry/borrow, negative, overflow, parity and zero flags as shown in Fig 7.

C	N	V	P	Z	11 bits empty for future use.

Fig. 7 Status Register

Individual bits of the status register are implicitly or explicitly read and/or written by the machine code instructions executing on the processor.

III. DESIGN OF RISC PROCESSOR

Attempts to achieve scalar and better performance have resulted in a variety of design methodologies that cause the CPU to behave less linearly and more in parallel. One of the simplest methods used to accomplish increased parallelism is to begin the first steps of instruction fetching and decoding before the prior instruction finishes executing. This simple technique known as instruction pipelining is utilized in almost all modern general-purpose CPUs [7]. Fig 8 shows the 3-stage pipeline structure used in this work.

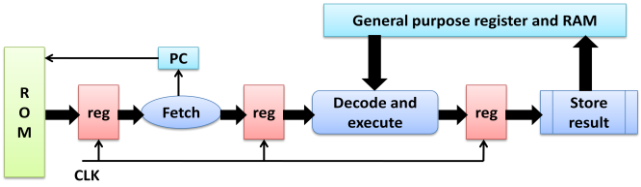


Fig. 8 3- Stage pipeline

The control unit (CU) coordinates the components of a computer system. It directs the operation of the other units by providing timing and control signals. All computer resources are managed by the CU. The control unit is a finite state machine that takes as its inputs the Instruction Register, the status register (which is partly filled by the status output from the ALU), and the current major state of the cycle. In the current design different combinatorial blocks like adders, multipliers, logic or shifters are selected depending on opcodes by control logic as shown in Fig 9. The corresponding registers from which data is to be fetched is also selected by CU.

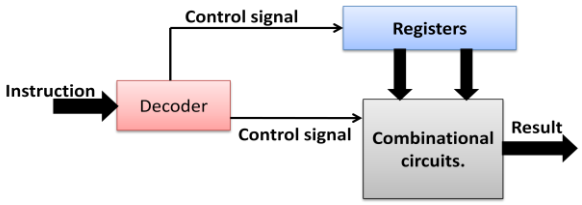
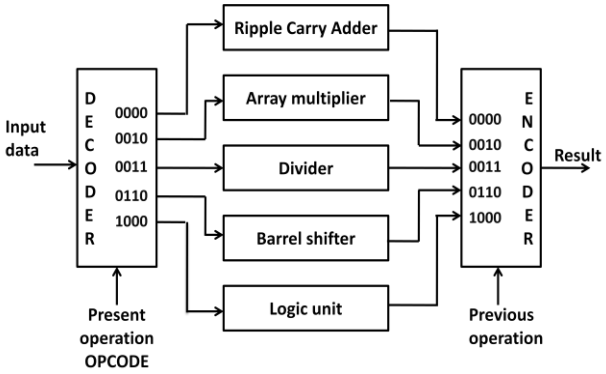


Fig 9. Control Unit

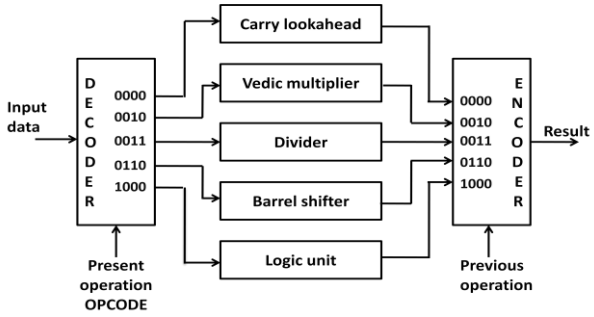
Fig 10 illustrates the details of the control unit which includes decoding the instruction and encoding the results on all three models[8][9].



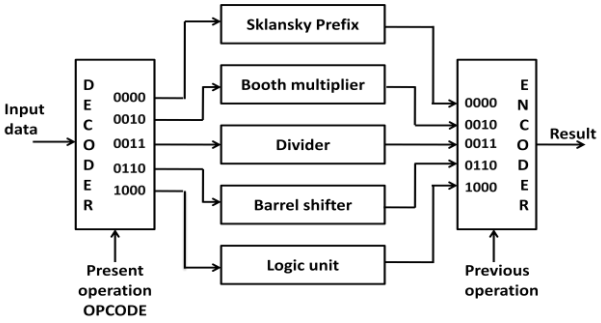
10(a)

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013



10(b)



10(c)

Fig. 10(a) –(c) Details of Control unit on all three ALU's

The opcodes that are designed have the structure as shown in Fig 11.

D15-D12	Operation
0000	ADD(with INC and DEC)
0001	SUB
0010	MUL
0011	DIV
0100	MOV
0101	
0110	SHIFT
0111	ROTATE

D15-D12	Operation
1000	LOGIC
1001	
1010	
1011	
1100	LOAD (IB)
1101	STORE(DB)
1110	CMP
1111	BRANCH

Fig. 11 Instruction Opcode Chart

A set of 8 general purpose registers from R0 to R7 have been used to aid in program execution. The assembled instructions are stored in memory. The detailed description of the instruction set designed is shown in Fig. 12.

INC & DEC (ADD &SUB)

0	0	0	0/1	DST(3 bit)	I	D	S	IN_REG(2)	IN_REG(1)

I --> Increment
D --> Decrement
S --> Sign flag ; all the flags are affected

MULTIPLY

0	0	1	0	DST(1)	DST(2)	IN_REG(2)	IN_REG(1)

DIVIDE

0	0	1	1	REM	QUOTIENT	IN_REG(2)	IN_REG(1)

IN_REG(1) must have default 8 bit data

Fig 12(a)

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

Since there are 8 registers, 3 bits of opcode are sufficient to address them. Since arithmetic operations need two input registers and one or two destination registers, 12 bits are set aside for the same. Wherever they aren't needed, special bits have been introduced.

SHIFT & ROTATE

0	1	1	0/1	DST(3)	d	t	LEN(4)	IN_REG

d → direction of rotation
0 --> right
1 --> left

t → type
0 --> logic
1 --> arithmetic

LOGIC

1	0	0	0	DST	OPERATION	IN_REG2	IN_REG1

OPERATION →
000 --> NOT
001--> AND
010--> OR
011--> XOR
100--> NAND
101--> NOR
110--> XNOR
111--> BIT RESET

Fig 12(b)

Since registers are 16 bit long, Len (4) indicates the number of bits to shift or rotate.

LOAD

1	1	0	0	M	U	I/D	REG (Pointer)(3) or immediate addr (6 bits)	IN_REG

M → 1 register addressing
→ 0 direct addressing

U → update

I/D → 1 increment pointing register
→0 decrement pointing register

STORE

1	1	0	0	M	U	I/D	REG (Pointer)(3) or immediate addr (6 bits)	OUT_REG

BRANCH

1	1	1	1	sub	mode	off	ret	Z	C	REG (3 bit) or immediate addr (6 bit) or offset(6 bit)

Sub → Sub routine.
Mode → 0 -> immediate addressing mode.
1 -> register addressing mode.
Off → offset (pc=pc + offset).
Ret → return statement.
Z → branch if zero flag equal to 1.
C → branch if carry flag equal to 1.

MOV

0	1	0	0	R	DST	REG (3 bit) or immediate data (8 bit)

R → 0 -> source is a register.
1 -> immediate data.

Fig 12(c)

Fig. 12 (a)-(c) Detailed Instruction Set

Branch instructions usually break the pipeline when control is taken away from usual course of instructions. To avoid this from happening, we have used yet another novel approach. We make use of the programming in Verilog to emulate a hardware which acts upon branch instructions separately, on finding them in the fetch stage. Thus further instructions are not fetched and latency or breakage is avoided.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

As seen from the flow-chart in Fig 13, on finding the branch instruction in the fetch stage, further fetching is stopped and based on the opcode of branch instruction suitable actions are performed.

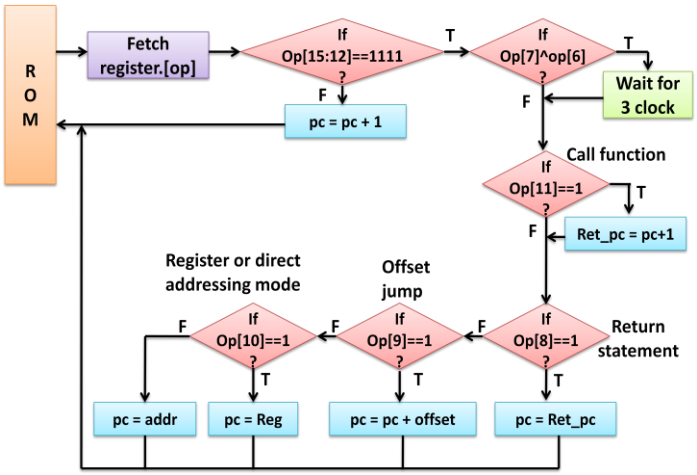


Fig. 13 Logic to avoid pipeline hazard (branch)

For instance, when the Offset mode bit is set, then PC is given that offset to jump to or if the instruction is of Return from subroutine type, then PC value is suitably changed. When branching is based on flags like carry and zero, then process is halted for 2 clock cycles so that the previous instruction can complete execution and corresponding flags updated. This structure is highly efficient in signal processing applications as lots of branch instructions are present.

IV. CONVOLUTION PROGRAM IMPLEMENTATION

The ALU is the core in DSP and ASIC where it is used in comparison, convolution, correlation, and digital filters. An ALU combines a variety of arithmetic and logic operations into a single unit. After designing an ALU, the necessity to verify and validate its performance is as such mandatory. DSP processors basically strive on filtering applications that need linear or circular convolution to be performed on inputs. As such these are RISC designs and hence we intend to implement Circular convolution on our processor .

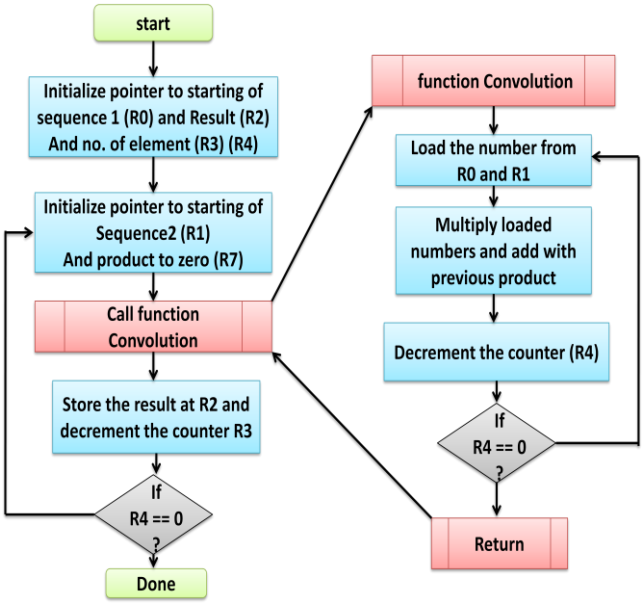


Fig 14. Circular convolution Flow chart

Fig.14 illustrates the algorithm used for convolution.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

Table. 1 Program of Circular Convolution (RISC)

MOV R0, 011b	0100 0 000 00000011
MOV R2, 10000b	0100 0 010 00010000
MOV R3, 3	0100 0 011 00000011
jmp: MOV R4, 3	0100 0 100 00000011
MOV R1, 0	0100 0 001 00000000
MOV R7, 0	0100 0 111 00000000
BL prod	1111 1 00000 010010
STRI R7, [R2]	1101 1 1 1 000 010 111
DEC R0	0000 000 0 1 0 000000
DEC R0	0000 000 0 1 0 000000
DECS R3	0000 011 0 1 1 000 011
BZ jmp	1111 0 0 0 0 1 0 000011
LDR R0, #10000	1100 0000 10000 000
LDR R0, #10001	1100 0000 10001 000
LDR R0, #10010	1100 0000 10010 000
LDR R0, #10011	1100 0000 10011 000
exit: B exit	1111 0 0 1 0 0 0000000
pro d : LDRI R5, R0	1100 1 11000 000 101
LDRI R6, R1	1100 1 11000 001 110
MUL R5,R6,R5,R6	0010 101 110 101 110
ADD R7, R7, R6	0000 111 000 111 101
DECS R4	0000 100 0 1 1 000 100
BNZ prod	1111 000 0 1 0 010010
RET	1111 0 0 0 1 00000000

Based on the algorithm, a RISC assembly program is written as shown in Table 1. The opcodes stored in the ROM, are fetched during execution. This program convolves any two sequences of 3 numbers.

V. FPGA IMPLEMENTATION AND RESULTS

The designs are implemented in Verilog HDL. For simulation and synthesis, Xilinx-ISE tool is used. Target device is SPARTAN-6 FPGA, based on 45 nm technology. In this project we use target technology and perform *place& route* operation for system verification. The inbuilt *Timing Analyzer* is utilized for speed based comparisons; X-power analyzer takes care of power analysis. Area of a design on FPGA is in terms of number of LUT's which is obtained from design summary.

Table. 2 Comparison of Adders

MODULE	DELAY (nS)	POWER (W)	LUTs
Carry Ripple	15.5	0.006	22
Carry Lookahead	14.07	0.008	60
Sklansky prefix	15.25	0.007	23

Table. 3 Comparison of Multipliers

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

MODULE	DELAY (ns)	POWER (W)	LUTs
Array multiplier	42.26	0.011	521
Booth multiplier	225.12	0.011	174
Vedic multiplier	28.01	0.009	488

Table 2 illustrates the comparison of specs of Adders designed, confirming high speed of CLA's. Table 3 illustrates comparison of specs on multipliers with booth occupying minimal area and Vedic being fastest.
The specs of the other blocks are given below:
Divider: The design has 125 LUTs, consumes 0.01W of power and delay of 112.6ns.
Barrel Shifter: The design has 68 LUTs, consumes 0.004W of power and delay of 11.4ns.
Dual Port RAM: Our design has just 32 words of memory, each 16 bit wide. The synthesized RTL has 512 registers and utilized 41 IOBs.
ROM: Our design had just 32 words of memory, each 16 bit wide. The synthesized RTL uses 8 LUT's and 22 IOBs.

Table 4. Final Specs based on Convolution

MODULE	NO. OF LUTs	POWER (W)	CLK PERIOD (ns)
ALU1	2267	0.01	11.837
ALU2	2872	0.009	10.216
ALU3	1462	0.012	10.847

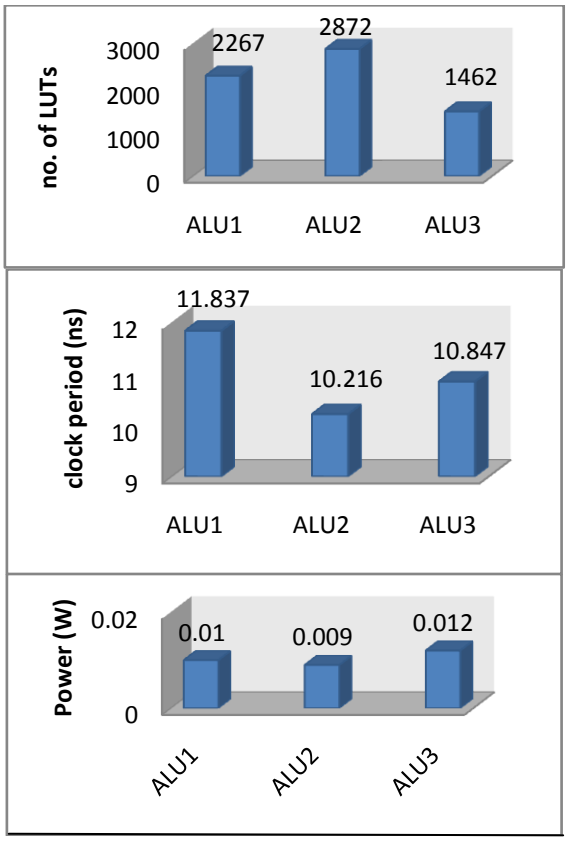


Fig. 15 Bar Graphs on Comparison of specs

The bar graphs of Fig 15 show a comparison of specs on the 3 ALU's.

VI. CONCLUSION AND SCOPE FOR FUTURE WORK

ALU1 is built in a simplistic way without tendering to power or delay or area constraints. Such structures are best made use in modules where performance in itself is not a priority but accuracy, pricing are. Low priced computing products, low cost controllers etc use them.
ALU2 (Vedic) is a generic high end product which caters to modern day demands of high speed, low power consumption and moderate area occupancy. Parallelism is best made use in these structures which has Carry-Lookahead adder and Vedic multiplier. High speed processors, mobile application processors and signal processing based devices (scopes), desktop PC's etc use them.
ALU3 is a tricky design. The focus here is to reduce area of the architecture compromising in both speed and power. Chip size is also a major concern in modern day designs since products scale down in size. Many applications desire such products where either of the three constraints has a higher preference over others. Sensor based applications are the best examples for these which serve for military and other purposes.
Thus we see how different structures are combined and varied spec results obtained. The combinations can be changed for sure, to obtain better results. For example prefix adders can be used as building blocks of multipliers which when used in the Vedic multiplier becomes a high speed and low power design which also tenders to minimal area requirements.
Going ahead, use of more complex prefix adders like Kogge-Stone adder can boost the performance. Also dividers based on algorithms like Newton Raphson, rather than sequential, can be utilized to achieve higher speed of execution. The next major step continuing the work would be to design a compiler for the processor and implement the design on a PCB. This would definitely require software expertise and also knowledge of the circuit design process.

REFERENCES

[1] Samir Palnitkar, "Verilog Hdl : A Guide to Digital Design and Synthesis" , Prentice Hall, 2nd Edition, 2003.
[2] B.D. Lee, V.G. Oklobdzija, "Improved CLA Scheme with Optimized Delay", Journal of VLSI Signal Processing, Vol. 3, p. 265-274, 1991.
[3] Mr. Abhishek Gupta, Mr. Utsav Malviya, Prof. Vinod Kapse, 'A Novel Approach to Design High Speed Arithmetic Logic Unit Based On Ancient Vedic Multiplication Technique', International Journal of Modern Engineering Research (IJMER) Vol.2, Issue.4, July-Aug 2012 pp-2695-2698 .
[4] Brent, R.P.; Kung, H.T. A regular layout for parallel adders. IEEE Transactions on Computers, vol.C-31, (no.3), March 1982. p.260-4.
[5] Fayez Elguibaly, 'A Fast Parallel Multiplier-Accumulator Using The Modified Booth Algorithm', IEEE Transactions On Circuits And Systems—II: Analog And Digital Signal Processing, Vol. 47, No. 9, September 2000.
[6] Charles H Roth Jr, "Digital Systems Design Using VHDL ", Prentice Hall 2nd Edition,
[7] Steve Furber, 'ARM- System On Chip Architecture', 2nd Edition , 2000.
[8] Samiappa Sakthikumaran, S. Salivahanan, V. S. Kanchana Bhaaskaran, "16-Bit RISC Processor Design for Convolution Application", IEEE-International Conference on Recent Trends in Information Technology, ICRTIT 2011, Anna University, Chennai. June 3-5, 2011.
[9] Vijay R. Wadhankar, Vaishali Tehre, " A FPGA Implementation of a RISC Processor for Computer Architecture", National Conference on Innovative Paradigms in Engineering & Technology (NCIPET-DEC 25,2012). Proceedings published by International Journal of Computer Applications.