Design and Implementation of Pipelined Reversible Vedic RISC Processor

[1]Adavipalli Dinesh,[2]Mr.P. Madhu Kumar,

[1]Student of M.Tech., [2]Assistant Professor(SL)
Department of ECE, Research Centre for VLSI and Embedded Systems
Sree Vidyanikethan Engineering College, Tirupati, India

Abstract: **This paper presents the design and implementation of a pipelined 16-bit RISC Processor. By optimizing the ALU circuit in RISC processor highly power efficient digital system can be achieved. The use of low power and high performance sub-blocks like adder and multiplier can reduce the total power dissipation of Reversible Vedic ALU. The various blocks include the Fetch, Decode, Execute and Store result to implement 4 stage pipelining. In RISC processor we are designing the reversible Vedic ALU. The only load and store is used to communicate with data memory. RISC exploitation pipeline makes CPI as one and improves speed of execution. Verilog Language is used for coding purpose. The proposed architecture is then simulated using Modelsim.**

Index Terms **- ALU, Reversible Logic Gates, Vedic Multiplier, RISC features, Pipelining, Divide and Conquer approach, HDL.**

I. INTRODUCTION

Verilog HDL has evolved as a regular hardware description language. A hardware descriptive language is a language used to describe a digital system. HDL's permits the design to be simulated earlier in the design cycle in order to correct errors or experiment with different architectures. Designs represented in HDL are technology independent, easy to design and debug, and are usually more readable than schematics, particularly for large circuits. More recently Verilog is used as an input for synthesis programs which will generate a gate-level description for the circuit. The simulator which is used for the language is Xilinx ISE and Modelsim [1]. Verilog is capable of describing simple behavior. Machine cycle instructions enable the processor to handle several instructions at the same time. The processor will work on a high clock frequency and thus yields higher speed. This paper is concerning design of a simple RISC processor and synthesizing it. The RISC architecture follows single-cycle instruction execution [2].

The main features of RISC processor are the instruction set is hardwired to speed instruction execution. In the present work, the design of a 4-bit data width Reduced Instruction Set Computer (RISC) processor is conferred. It has a complete instruction set, program and data memories, general purpose registers and a simple Arithmetical Logical Unit (ALU) for basic operations. In this design, most instructions are of uniform length and similar structure, arithmetic operations are restricted to CPU registers and only separate load and store instructions access memory. The architecture supports 8 instructions to support Arithmetic, Logical, Shifting, and load -store operations.

Reduced Instruction Set Computers (RISCs) are now use for all type of computational tasks. In the area of scientific computing, RISC workstations are being increasingly used for compute task such as DSP, DIP etc. RISC concepts help to achieve given levels of performance at significantly lower cost than other systems. Pipelined RISC improves speed and cost effectiveness over the ease of hardware description language programming and conservation of memory and RISC based designs can continue to grow in speed and ability.

II. RISC Processor Architecture

RISC design has been developed as a result of the 801 project which started in 1975 at the IBM T.J.Watson Research Center and was completed by the early 1980s. RISC design starts with a little set of most frequently used instructions which determine the pipeline structure of the machine enabling fast execution of those instructions in one cycle. The IBM was the first company to define the RISC (Reduced Instruction Set Computer) architecture in the 1970s. This research was further developed by the universities of Berkeley and Stanford to grant basic architectural models [3].

A. CISC v/s RISC

Processors have conventionally been designed just about two Philosophies: Complex Instruction Set Computer (CISC) also Reduced Instruction Set Computer (RISC). The CISC model is a move toward the Instruction Set Architecture (ISA) design so as to emphasize responsibility extra through every Instruction by an extensive selection of Addressing modes; the Instructions are of broadly unreliable lengths along with execution times therefore challenging a very complex Control Unit, which tends to large chip area.

On the other hand, the RISC Processor works on reduced number of Instructions, fixed Instruction length, further general-purpose registers, load-store architecture along with simplified Addressing modes which makes individual instructions execute faster, achieve a net gain within performance as well as an generally simpler design through less silicon consumption because compared toward CISC.

Currently RISC is one of main significant tool in computer systems for the reason that of following points,

- RISC instructions are executed within single clock cycle, as the majority of CISC require further than one clock cycle.
- RISC system is more popular.
- The technology RISC processors carry the floating point data type.
- RISC machines generally utilize hardwired unit.
- RISC machines need lesser time designed for its design implementation.
- RISC processor consume less power
- RISC machines utilize load as well as store instruction to access data on or after memory.

B. RISC features

The key features of RISC processor are the instruction set can be improved to speed instruction execution. No microcode is desirable for single cycle execution. The entire instructions are fixed bit in length. This simplifies the instruction fetch device as the position of instruction boundaries is not a function of the instruction type. The processor has small number of addressing modes. Simply load also store instructions access memory, load/store instructions operate among memory also a register. The machine cycle time is minimized. The fixed dimension of the instructions allows the instructions on the way to be simply piped. RISC provides a flexible as well as expandable architecture to maximize performance as of every known semiconductor technology. RISC includes extensions toward RISC concepts to facilitate complete known levels of presentation by extensively lower cost than further systems. The design procedure is extremely fast also cost effective. In this design, mainly instructions are of identical length also similar structure, arithmetic operations are limited toward CPU registers moreover simply separate load also store instructions access memory . The Instruction cycle consists of four stages specifically fetches, decode, execute as well as Store. After each instruction fetch, Control Unit generate signals used for the selected Instruction. The architecture supports 8 instructions toward carry Arithmetic, Logical, Shifting also Load-store operations.

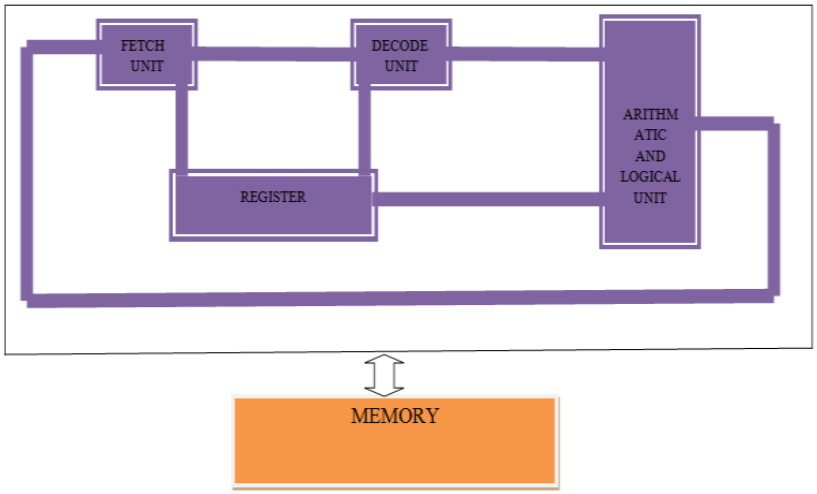

Figure 1: Block diagram of RISC processor

The Block Diagram of a 16 - bit RISC processor is exposed within Figure1. The RISC processor architecture consists of Arithmetic Logic Unit (ALU) (including arithmetic along with logical units, barrel shifter, vedic multiplier), control unit (CU), Memory as well as Register File. RISC processor is designed among load/store architecture, meaning that all operations is performed on operands detained in the processor registers along with the main memory can only be accessed through the load and store instructions [4].

C. Pipelining

Instruction and data are fetched in sequential order consequently to the latency incurred among the machine cycles can be reduced. For increasing the speed of operation RISC processor is designed with four stage pipelining. The pipelining stages are Instruction Fetch (IF), Instruction Decode (ID), and Execution (EX) also Store result (ST).

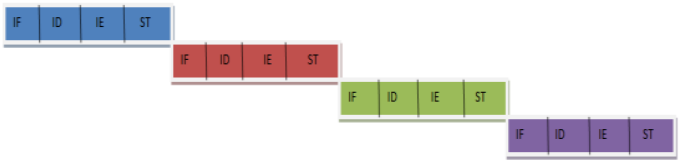

Figure 2: instruction execution without pipelining technique

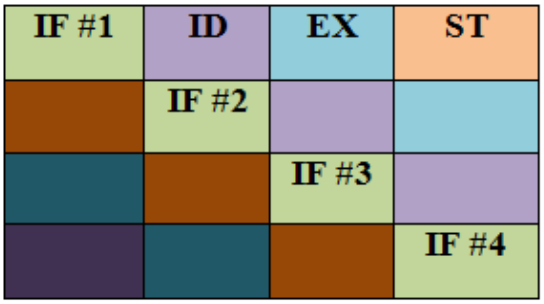

Figure 3: Instruction execution without pipelining technique

The Figure2 shows the execution of instruction without taking concept of the pipeline. Then there will be extended time necessary intended for processor to execute set of instruction. Consequently CPI will be very high.

The Figure3 shows the execution of instruction as a result of by the method called pipelining. Execution time required for processor can be reduced as well as speed of execution increases. Thus CPI can reduce. Let us take every step of pipeline method like,

Instructions fetch cycle (IF): Send the program counter (PC) to memory and fetch the current instruction from memory . Update the PC to the next sequential PC through adding one toward the PC.

Instructions decode/register fetch cycle (ID): Decode the instruction also read the registers corresponding toward register source specifies from the register file. Decoding is done into parallel through reading registers, which is possible since the register specifies, are by a fixed location in RISC architecture. This method is known while fixed-field decoding.

Execution (EX): The ALU operates on the operands prepared in the prior cycle, performing one of three functions depending on the instruction type. Register-Register ALU instruction: The ALU performs the process specified through the ALU opcode on the values read from the register file.

Store result (ST): Register-Register ALU instruction otherwise Load instruction: Write the result into the register file, whether it comes from the memory system (for a load) or else from the ALU (for an ALU instruction).

The purpose of the instruction fetch unit is to obtain an instruction from the instruction memory by the current value of the Program counter (PC) along with increment the PC value used for the next instruction. The major role of the instruction decode unit is to utilize the 16-bit instruction provided from the previous instruction fetch unit to index the register file also obtain the register data values. The instructions opcode field's bits are sending toward a control unit to determine the type of instruction to execute. The type of instruction that decides which control signals are to be set along with function that Execute unit is to perform, thus decoding the instruction.

D. Main modules

Let us consider modules of RISC processor. The main parts of processor is shown in Figure 4 and are explained below

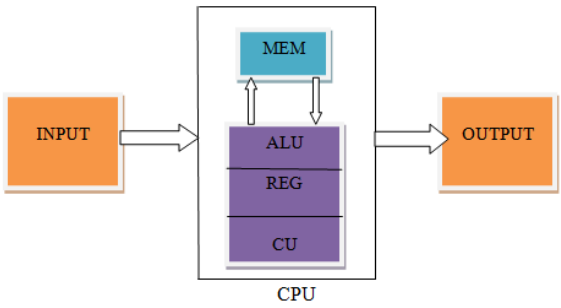

Figure 4: Basic main modules of processor

These are following basic major parts of the processor:

Control Unit: It manages the sequence with timing of events carried out within the processor. The control unit of the RISC processor examines the instruction opcode bits also decodes the instruction to generate eight control signals.

Registers: Holds values of internal operation, such as the address of the instruction being executed along with the information being processed i.e. Program Counter Register, Status Register.

Separate program and data memory: The program memory also called as ROM which contains instructions of the processor. Every instruction is 16 bit also there are 8 instructions are there. The data memory which also called as RAM which is temporary memory also used to store the data values need for processor as well as data values coming from processor after processing.

Load and store: Processor which communicates through memory simply by using load as well as store instruction. Load instruction which load the data value from memory to register also store instruction which store the value from register to RAM memory.

Arithmetic Logic Unit (ALU): The arithmetic/logic unit (ALU) executes every arithmetic as well as logical operations. Arithmetic operations also obtain two registers as operands. The result is stored in a third register. The arithmetic/logic unit can perform arithmetic operations or else mathematical calculations like addition, as well as subtraction along with also performs logical operations contain Boolean comparisons, such as AND, OR, XOR, NAND, NOR along with NOT operations. Mostly within RSIC processor we designed reversible Vedic ALU

The block diagram of proposed ALU is shown in Figure.5 which consists of an arithmetic unit, logic unit along with a multiplexer. Arithmetic unit is designed to perform the arithmetic operations like addition, subtraction, multiplication also division. Logic unit is intended to execute the logic operations using basic logic gates through reversible logic. Multiplexer is used to select any one of the circuit operation also arithmetic or else logic operation along with directs the output of respective unit to the output terminals of ALU. The circuit design as well as implementation of sub-blocks is described in detail below.

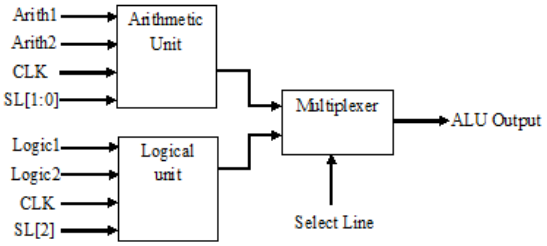

Figure 5: Block Diagram of proposed ALU

Arithmetic Unit: The logic blocks of arithmetic unit are intended with reversible logic gates as well as Vedic multiplier [5] which are used to reduce the power dissipation as well as transistor count of arithmetic unit. The block diagram of arithmetic unit is shown in Figure.6. The reversible logic gate is employed to obtain the logic operation through low power dissipation [6].

III. Reversible logic gate

The logic block of a proposed reversible logic gate is shown in Figure.7 in which the reversible logic gate accepts three inputs as well as provides three outputs. One to one mapping method is used in the reversible logic gate that helps to determine the outputs from input and that outputs can be used to recover the inputs . It satisfies the demand of Landauer's principle as well as C.H. Bennett rule that are defined for energy efficiency [7]. The reversible gate can be used to perform all the logic gates through high speed also low area along with power dissipation.

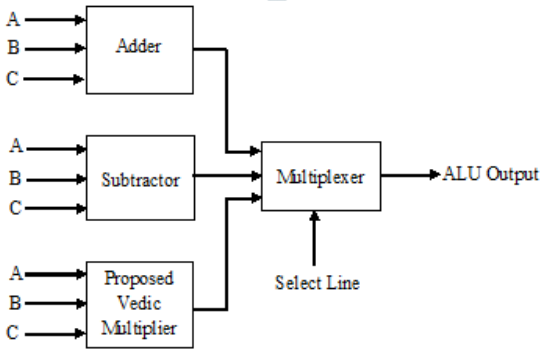

Figure 6: proposed Arithmetic Unit

Reversible logic gate can be used to perform logical operations like AND, OR, NAND, NOR, Buffer, XOR, XNOR through high speed, less area along with less complexity. In this logic by any immediate of instance one of the input acts as a control input along with other two works seeing that data input. The logic circuit computation using reversible logic gate improves the efficiency of the circuit through low power dissipation [8].

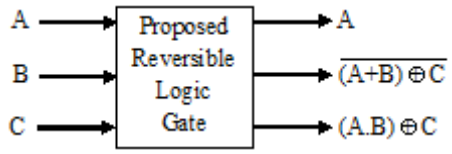

Figure 7: proposed Reversible Logic Gate

IV. Vedic Multiplier

Frequently multipliers are implemented in the digital system using multiple numbers of adders along with such a multipliers are implemented through the help of array along with booths algorithms. But Vedic mathematics illustrates the multiplication operation very faster than other algorithms by reducing number of addition. The reduced number of adders in Vedic multiplier minimizes the operating delay along with area of device by less number of adders [9]. For example consider a N × N multiplier, it may need N× (N-2) full adders, N half-adders as well as N×N AND gates to execute the multiplication. Except Vedic multiplier requires simply N¬2 gates as well as (N–1) adders. Because the use of less number of adder Vedic multiplier occupy less area along with consumes reduced power in logic implementation along with operation respectively.

A proposed 4-bit Vedic multiplier is shown in Figure.8 in which the logic evaluation of multiplication procedure is performed using divide also conquer method. The logic multiplication of two 4-bit binary numbers through proposed Vedic multiplier is described below. Let us imagine the binary inputs are A4A3A2A1 also B4B3B2B1. The output expressions of proposed Vedic multiplier are

$$X0[3:0] = A[1:0] \times B[1:0]$$
$$X1[3:0] = A[1:0] \times B[3:2]$$
$$X2[3:0] = A[3:2] \times B[1:0]$$
$$X3[3:0] = A[3:2] \times B[3:2] \quad (1)$$

After that the logic addition of multiplied terms are performed as follows

$$Y[1:0] = X0[1:0]$$
$$Y[2] = X0[2] + X1[0] + X2[0] + 1'b0$$
$$Y[3] = X0[3] + X1[1] + X2[1] + 1'b0$$
$$Y[4] = X1[2] + X2[2] + X3[0]$$
$$Y[5] = X1[3] + X2[3] + X3[1]$$
$$Y[7:6] = X3[3:2] \quad (2)$$

In this proposed logic the logic addition is performed with reversible logic gates which are an energy efficient arithmetic logic circuit in the digital system design . The logic output of 4-bit Vedic multiplier is designed through the combinations of 2-bit Vedic multipliers. The block diagram of 2-bit Vedic multiplier is shown within Figure 9. It is implemented with the adders as well as logic gates [10].

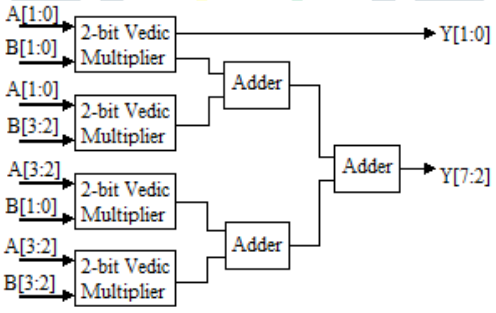

Figure 8: proposed 4-bit Vedic Multiplier

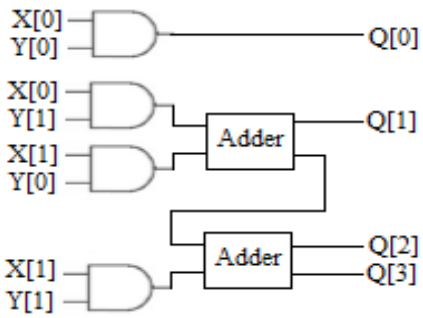

Figure 9: proposed 2-bit Vedic Multiplier

V. Logical Unit

The block diagram of logic unit is shown within Figure.10 which is consisting of a reversible logic gate also a multiplexer. The logic outputs of reversible gate are selected by the multiplexer. The energy efficient reversible logic gate is used to perform the logic operations through reduced power dissipation and area [11].

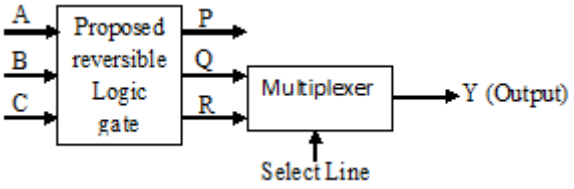

Figure 10: Block diagram of proposed Logical Unit

VI. RESULTS AND DISCUSSION

Block Diagram

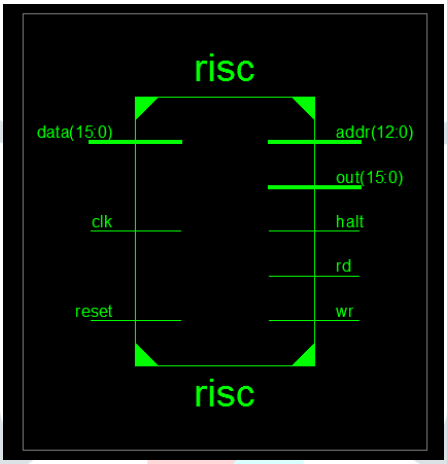

Figure 11: Block Diagram for RISC Processor

RTL Schematic

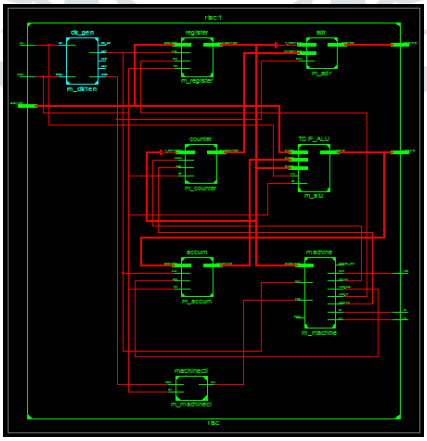

Figure 12: RTL Schematic for RISC Processor

The above Figure shows the RTL schematic of the Design. In this Design we can observe the internal modules of the Design.

SIMULATION RESULTS

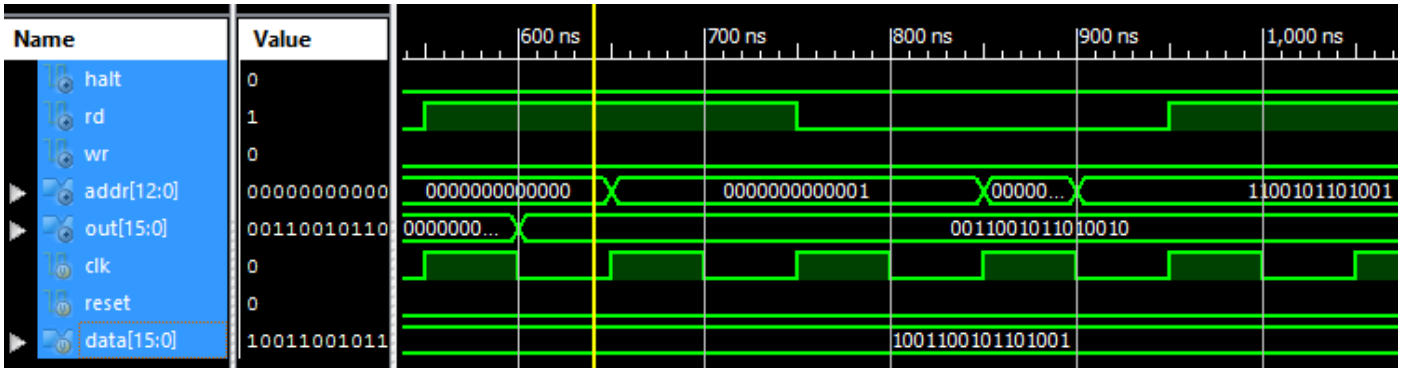

Figure13: Simulation Results For RISC Processor

The above image shows the simulation result of the proposed design. Clk and rst are the control inputs and data is the data input. Out is the data output and rd,wr indicates whether read operation has to be done or write operation. Addr shows where the data output to be located and halt shows whether the process terminated or still running.

TABLE 1. COMPARISON OF CONVENTIONAL RISC PROCESSOR WITH PROPOSED RISC PROCESSOR

SPECIFICATIONS	EXISTING	PROPOSED
No. of slice	1122	634
No.of 4 input LUTS	2070	1182
No.of Slice FF	83	83
Delay	75.654ns	74.225ns

From this comparison Table, the proposed RISC processor Design proves to be a better choice in terms of Area & Delay.

VII. CONCLUSION

In this paper the 16 bit RISC Processor core has been design and simulated in Xilinx ISE 14.5. The design has been achieved using Verilog and simulated with Modelsim simulator. Most of the goals were achieved and simulation shows that the processor is functioning perfectly. Every instruction is executed in one clock cycles with 4-stage pipelining. The design is verified through simulations. Also the proposed low power and high performance ALU circuit is implemented. The low power and area efficient reversible logic based adder and Vedic multipliers are integrated to construct architecture of ALU which resulted in low power dissipation in comparison with the conventional unit. The use of less number of adders in the Vedic multiplier reduces the area and power dissipation of ALU system. The synthesized output shows that the proposed technique reduces area and delay when compared to conventional RISC processor.

.

References

[1] Rakesh M.R " RISC Processor Design in VLSI Technology Using the Pipeline Technique " INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING Vol. 2, Issue4, April 2014.

[2] Galani Tina G., Riya Saini and R.D.Daruwala, July-2013, "Design and Implementation of 32 – bit RISC Processor using Xilinx"

[3] Samiappa Sakthikumaran ,S.Salivahanan and V.S.Kaanchana Bhaaskaran , June 2011, "16-Bit RISC Processor Design For Convolution Application" ,IEEE International Conference on Recent Trends In Information Technology, pp.394-397.

[4] R. Uma, Mar-Apr 2012, "Design and Performance Analysis of 8-bit RISC Processor using Xilinx Tool"

[5] Senthil Sivakumar M, Sundaram A, Gurumekala T, Banupriya M " Design of ALU using reversible logic based

Low Power Vedic Multiplier " International Journal of Scientific & Engineering Research, Volume 6, Issue 2 , February-2015.

[6] Mahapatro M, Panda, S.K. ;Satpathy, J. ; Saheel, M, "Design of Arithmetic Circuits Using Reversible Logic Gates and Power Dissipation Calculation", Electronic System Design (ISED), IEEE 2010, pp.85-90.

[7] Maity M, Ghosal P, Das B, "Universal reversible logic gate design for low power computation at nano-scale" Microelectronics and Electronics (PrimeAsia), IEEE 2012, pp.173-177.

[8] Bruce, J.W, Thornton, M.A, Shivakumaraiah, L, Kokate, P.S, "Efficient adder circuits based on a conservative reversible logic gate", VLSI 2002, IEEE Proceedings, pp.74-79.

[9] Kunchigi V, Kulkarni L, Kulkarni S, "High speed and area efficient Vedic multiplier", Devices, Circuits and Systems (ICDCS), IEEE-2012, pp.360-364.

[10] Morrison M, Ranganathan N, "Design of a Reversible ALU Based on Novel Programmable Reversible Logic Gate Structures", VLSI, IEEE 2011, pp. 126- 131.

[11] Rakshith T.R, Saligram, Rakshith, "Design of high speed low power multiplier using Reversible logic: A Vedic mathematical approach", ICCPCT, IEEE 2013, pp.775-781.