European Journal of Molecular & Clinical Medicine
ISSN 2515-8260 Volume 7, Issue 4, 2020

Implementation of a 32-bit MIPS based on the RISC processor by using VHDL

[1]Sateesh Amarneni, [2]Mallesh Sudhamalla, [3]Dr.K.Bharat Kumar, [4]Dharma Teja Sabbani, [5]Uma Boda

[1]Vidya Jyothi Institute Technology,Hyderabad,India,500075.
[2]CMR Technical Campus, Hyderabad,India
[3]CMR Technical Campus,Hyderabad,India
[4]Vidya Jyothi Institute Technology,Hyderabad,India,500075.
[5]Vidya Jyothi Institute Technology,Hyderabad,India,500075.
sateeshaece@vjit.ac.in

ABSTRACT: In this document a simple 32 bit RISC based MIPS type processor by using Peres Reversible Logic Gates, which gives transaction in delay and frequency. The synthesis and simulation is conceded out by using XILINX ISE 14.5 and HDL is urbanized by means of VHDL language with vertex device. The smart world looking in making of smart gadgets by using processors and decade by decade the speed and other parameters are getting minimized for better performance. here introducing to RISC based MIPS architecture based processor for high speed is done .which works without compiler where the existed general processor is in use of .with the use of MIPS RISC architecture the arithmetical, logical and shift functions are done in reduced time as there is reduced instruction set computer, it employs a good number of registers than to transistors. The processor internally consists of some blocks so that the operations are performed. RALU(reversible gate arithmetic logic unit) is considered to be the main module of the processor which provides task functions and other blocks are also exists such as control unit ,program counter ,instruction register ,memory .

Keywords: *RISC, RALU, MIPS, FCU, XILINX.*

1. INTRODUCTION

The intend of 32-bit RISC based MIPS processor employ among a mixture of blueprint blocks like RALU, Accumulator, Program Counter, Instruction Register, Memory, Control Unit, and additional logic. The intend integrate several the subsequent issues which are based on 32 bit of data and 28 bit of address. It seize a predetermined tuition format which is of length 32 bit, and range of Opcode is of size 4 bit, handling 15 instructions which has 256 memory locations, 32-bit registers (IR, ACC), implement a 2-staged pipelining which overlap of fetch and execute cycles, No interrupts and No provisional twigs, Data that it handle a mysterious integer type.

- Complex instruction set microprocessor –

The processors are focused on the amount of instructions per program to a certain extent than number of cycles per commands. The compiler employ in paraphrase of a high level language to assembly level language where the code length is moderately miniature where extra RAM is relay to accumulate the instructions. Von Neumann architecture intended by the named physicist and mathematician John Von Neumann in the late 1940s,.for dropping the putting to death time the CISC based micro processor is replaced by RISC based micro processor in which enticing and implementation are finished in dissimilar stages.

- Reduced instruction set microprocessor –

These processors are intended as per function necessity. These are engaged to reduce the execution time by means of the simplified instruction set.These processors implement commands at faster rate. They process in only one clock cycle in implementing a result at uniform execution time. There are good number of registers and low number of transistors. Harvard architecture which was based on the original Harvard Mark I relay-based computer which employed separate memory systems to store data and instructions.

2. ARCHITECTURE OF RISC BASED MIPS PROCESSOR

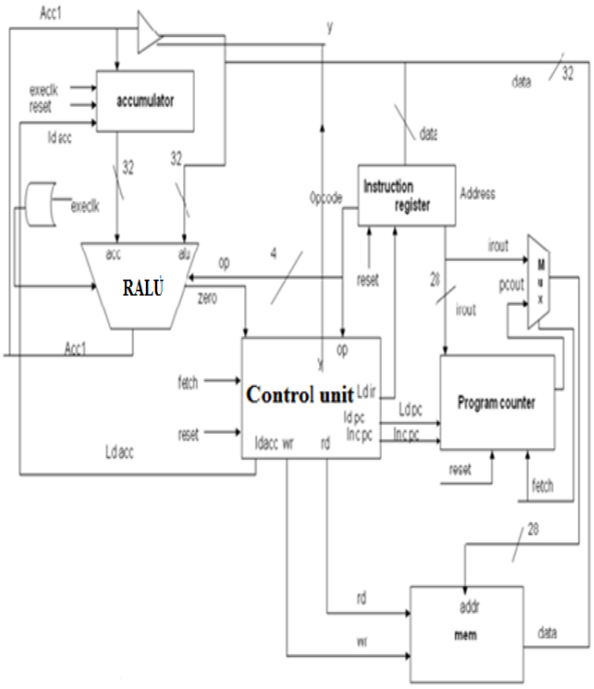

Fig:1 Architecture of RISC based MIPS processor
Internal blocks

Accumulator:

Accumulator is used to store the output value of the ALU and again driven as one of the input which the operation is done for the previous stored value and present value based on opcode .the opcode is of 4 – bit size. It employs as the primary register which stores the previous output.

The result of an ALU operation is forever store in accumulator at a few particular time based on the control logic instruction and also the external clock. This output is another time feed to ALU as input. If Reset =0, the output of accumulator is unfurnished to zero. When reset is high and load accumulator signal is set high, the output of the ALU is burdened in to the accumulator at the negative edge of the external clock.

Instruction register:

Instruction register provides the instruction in the opcode format so that internal ALU process is going to perform instruction related operations with the present data and previous data as inputs for that particular operation.

Memory:

The 256 bit storage ram is used to storing and loading of data and is done by the write and read signals. The data storage is decided by the control unit .which is said to perform the control operations and the particular address location is decided by the multiplexer that has program counter output and instruction register output as inputs and a selection signal of fetch is employed.

Program Counter:

The program counter is the increment counter in which the next instruction address is stored as per the increment pc input provided by the control signal in same clock time as the previous instruction execution is in process.

RALU:

The RALU perform both arithmetic and logical operations by using the reversible gates and as glowing as organize with transfer instructions. It takes data and acc as inputs to generate output according to the opcode. An external clock is given as input for synchronization and the output is accessible at positive edge of the external clock. It perform arithmetic and logic instructions directly and control of transfer instructions are performed with the facilitate of control and logic decoder.

Control unit:

The control unit controls all other blocks and notifies the functioning of the blocks. It has fetch and opcode as inputs and

Load program counter,inc program counter, load instruction register, read memory ,write memory will be the outputs which are driven to their respective blocks as inputs.

3. WORKING OF RISC BASED MIPS PROCESSOR

The 32 bit RISC processor is developed to perform logical arithmetic and control operations with minimized execution time by employing the opcode as instruction by which the predefined instructions are applied based on its particular opcode match.

These are done in the ALU in which predefined instructions are coded with a particular representing address and selection of the instruction is carried by providing input instruction address .The operations are performed as per the opcode based instruction and the resultant of RALU is stored in the accumulator and in the next execution cycle the ALU is provided the accumulator stored value as one of the input and again the output is stored in accumulator.

The operations are done with the operands as previous resultant and present data which forms a sequential circuit and is performed based on the external clock .The control unit is employed with fetch and opcode input by which it controls the incrimination of pc and load instruction register and read or write the memory is done within .which can be considered as the main functional unit of the processor.

As the control unit outputs are driven as inputs to the remaining blocks such as for memory write and read signals and for program counter incpc and ld pc and for instruction register ldir. The instruction register is of 32 bit register in which first 4bits from msb are classified as opcode and remaining bits as the address and having output of irout. The program counter haves input of fetch along with control unit outputs and gives output of pc out which is of 32 bit

These instruction register output and pc out are driven into the multiplexer block in which the selection is done based on the fetch signal and the multiplexer output is used as the address location of the input data to be stored in 256 bit size ram memory. The buffer is employed and set as high so that if the looping error while occurring zero can be overcome and make the error loop terminate in multiplication and many other logical operations

4. REVERSIBLE LOGIC GATES

A reversible logic gate is a type of n-input n-output logic device with one-to-one mapping.From the point of reversible circuit design, there were many parameters to determine the involution and performance of circuits. In this, an ameliorated design of reversible multiplier with deference to its anterior counterparts is proposed. Multiplier circuits play a consequential role in computational operation utilizing computers. There are many arithmetic operations which are performed, on a computer FCU, through the utilization of multipliers.

Feynman Gate

Figure 3 shows a 2*2 Feynman gate . Quantum cost of a Feynman gate is 1.Feynman gate is called as Controlled NOT gate or CNOT gate. It is equivalent to single control input toffoli gate.

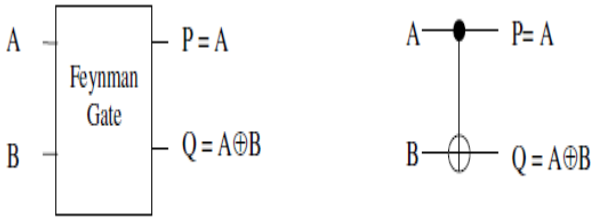

Fig.2: feynman gate and its symbolic representation.

Toffoli Gate:

Figure 4 shows a 3*3 Toffoli gate The input vector is I(A, B, C) and the output vector is O(P,Q,R). The outputs are defined by P=A, Q=B, R=A(B xor C). Quantum cost of a Toffoli gate is 5. It has two control inputs.

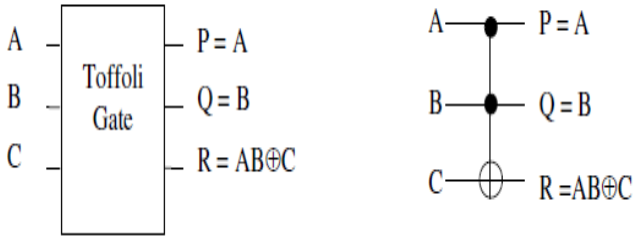

Fig.3: Toffoli gate and its symbolic representation.

Peres Gate:

Figure 5 shows 3*3 Peres gate. The input vector is I (A,B,C) and the output vector is O (P,Q,R). the output is defined by P=A, Q= A^B and R= A&B ^C. quantum cost of a Peres is 4. It needs two Toffoli gates for its construction.

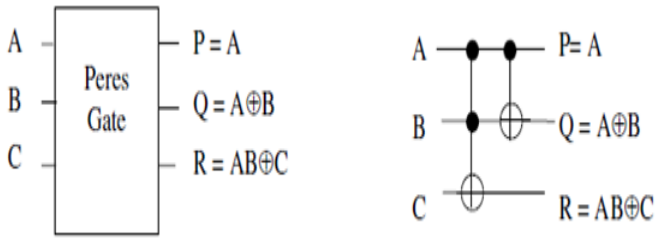

Fig.4: Peres gate and its symbolic representation.

CNOT GATE:

CNOT gate is also known as controlled-not gate. It is a 2*2 reversible gate. The CNOT gate can be described as:

Iv = (A, B) ; Ov = (P= A, Q= A B)

Iv and Ov are input and output vectors respectively.m Quantum cost of CNOT gate is 1. Figure shows a 2*2 CNOT gate and its symbol.

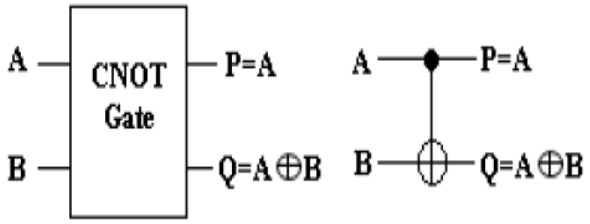

Fig.5: CNOT gate and its logic symbol.

NFT Gate:

It is a 3x3 gate and its logic circuit and its quantum implementation is as shown in the figure. It has quantum cost five

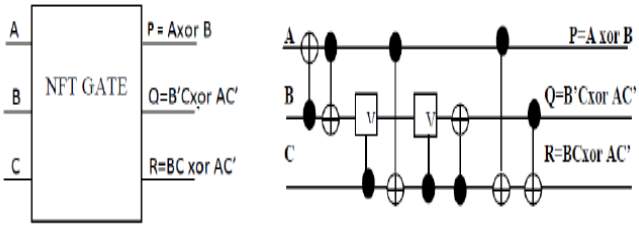

Fig.6: NFT gate and its Logic symbol.

RALU

RALU consists of the design architecture based on reversible logic gates any operation is carried out by the adder unit so the design of adder is designed by using the peres gate by which the power consumption and temperature tradeoff can be reduced so that enhanced results can be noticed.construction of the full adder with using peres gate is shown below

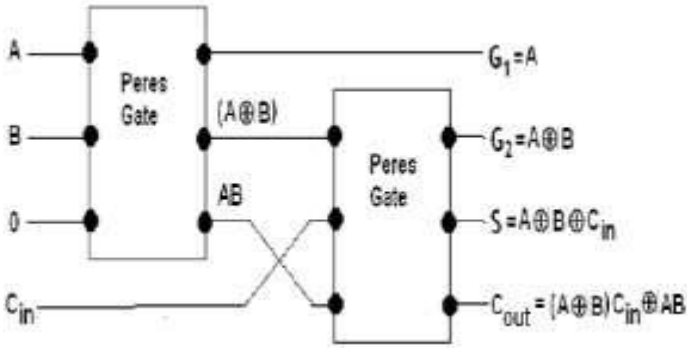

Fig :7 Full adder with two peres gates

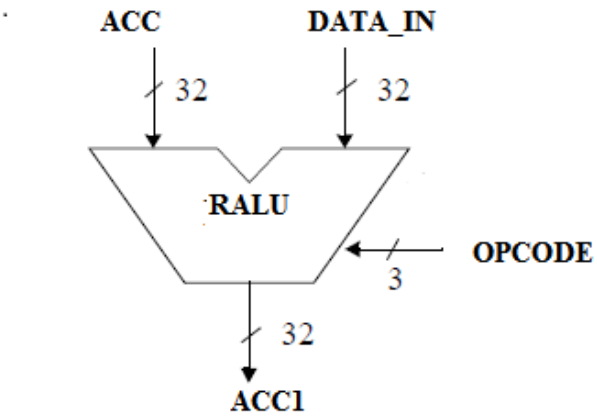

Fig:8 Block diagram of RALU

5. RESULTS

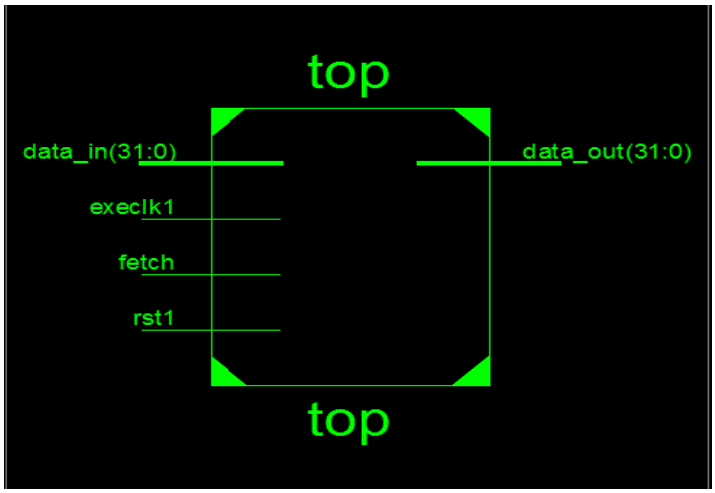

Fig:9 RTL schematic

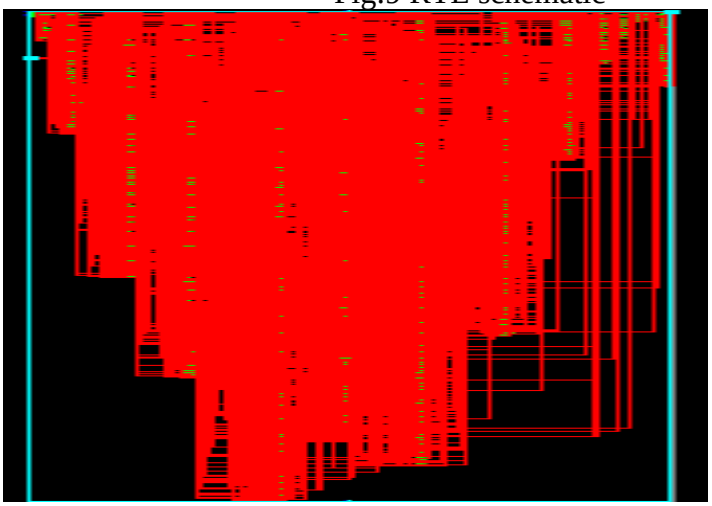

Fig:10 Technology schematic view

6. CONCLUTION

RISC authentically enhances the performance of processor by considering the factors like simple architecture construction and ordinant dictation set, facile injuctive authorization set for decoding and simplified control architecture. For performing these operations, this processor contains the major blocks as Control unit (CU), ALU; Program counters (PC), Accumulator, Ordinant dictation register, Recollection and adscititious logic. The MIPS based RISC processor got the trade off with minimum delay of 8.36ns which is said to be having reduced time period so that the operations can be perform more quicker with the maximum frequency of 119.388 ..hence the MIPS based RISC achieves the uktimate target of less dealy by which speed can be boosted. The results are scribbled by using vertex device in Xilinx ise 14.5 in vhdl language.

7. REFERENCES

[1] P. Ienne and R. Leupers, *Customizable Embedded Processors: Design Technologies and Applications*. San Francisco, CA, USA: Morgan Kaufmann, 2007.

[2] P. M. Heysters, G. J. M. Smit, and E. Molenkamp, “A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems,” *J. Supercomput.*, vol. 26, no. 3, pp. 283–308, 2003.

[3] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix," in *Proc. 13th Int. Conf. Field Program. Logic Appl.*, vol. 2778. 2003, pp. 61–70.
[4] M. D. Galanis, G. Theodoridis, S. Tragoudas, and C. E. Goutis, "A high-performance data path for synthesizing DSP kernels," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 6, pp. 1154–1162, Jun. 2006.
[5] K. Compton and S. Hauck, "Automatic design of reconfigurable domainspecific flexible cores," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 5, pp. 493–503, May 2008.
[6] S. Xydis, G. Economakos, and K. Pekmestzi, "Designing coarse-grain reconfigurable architectures by inlining flexibility into custom arithmetic data-paths," *Integr., VLSI J.*, vol. 42, no. 4, pp. 486–503, Sep. 2009.
[7] S. Xydis, G. Economakos, D. Soudris, and K. Pekmestzi, "High performance and area efficient flexible DSP datapath synthesis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 3, pp. 429–442, Mar. 2011.
[8] G. Ansaloni, P. Bonzini, and L. Pozzi, "EGRA: A coarse grained reconfigurable architectural template," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 6, pp. 1062–1074, Jun. 2011.
[9] M. Stojilovic, D. Novo, L. Saranovac, P. Brisk, and P. Ienne, "Selective flexibility: Creating domain-specific reconfigurable arrays," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 5, pp. 681–694, May 2013.
[10] R. Kastner, A. Kaplan, S. O. Memik, and E. Bozorgzadeh, "Instruction generation for hybrid reconfigurable systems," *ACM Trans. Design Autom. Electron. Syst.*, vol. 7, no. 4, pp. 605–627, Oct. 2002.
[11] [Online]. Available: http://www.synopsys.com, accessed 2013.
[12] T. Kim and J. Um, "A practical approach to the synthesis of arithmetic circuits using carry-save-adders," *IEEE Trans. Comput.- Aided Design Integr. Circuits Syst.*, vol. 19, no. 5, pp. 615–624, May 2000.
[13] A. Hosangadi, F. Fallah, and R. Kastner, "Optimizing high speed arithmetic circuits using three-term extraction," in *Proc. Design, Autom. Test Eur. (DATE)*, vol. 1. Mar. 2006, pp. 1–6.
[14] A. K. Verma, P. Brisk, and P. Ienne, "Data-flow transformations to maximize the use of carry-save representation in arithmetic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1761–1774, Oct. 2008.