Advanced Low Power RISC Processor Design using MIPS Instruction Set

P.V.S.R.Bharadwaja [1] ,*K. Ravi Teja* [2], *M.Naresh babu* $_{M.E.}$ [3] ,*K.Neelima* $_{M.Tech}$ [3]
[1] P.G.Student (VLSI) , Dept. of E.C.E. , S.V.E.C. , Tirupati , A.P.,India
[2] P.G.Student (VLSI) , Dept. of School of Electronics. , V.I.T., Vellore , T.N. , India
[3] Assistant Prof. , Dept. of E.C.E.,S.V.E.C., Tirupati , A.P., India

***Abstract*—Present era of SOC's comprise analog, digital and mixed signal components housing on the same chip. In this environment processor plays a vital role. As the technology shrinking to sub-micrometer technology node, there exists a huge scope of undesirable hazards in processors. These hazards may lead to disturbance in area, power and timing which deviate from desired quantities. Our paper focuses mainly to solve some of these issues. In-order to tackle these problems, we are introducing the enhanced version of MIPS. Microprocessor without Interlocked Pipeline Stages (MIPS) is a recent architecture into the semi-conductor industry. This paper totally concentrates on designing the architecture in Verilog HDL. The design had been simulated and synthesized in Nc-launch and RTL-compiler licensed by cadence Inc respectively. The physical design of synthesized architecture had been carried on by Socencounter under slow.lib library of TSMC Cmos 180nm technology node.**

.

Keywords—Hazard Detection Units , Low Power Processor , MIPS, RISC using MIPS

I. INTRODUCTION

Processor is the crux of the computer. There are aggregate amount of processor's in the market. Customarily a lot many new amount of Processor's are also into the market. Out of all these processor's a few of them were designed using processor cores i.e. Hardware Description Languages like Verilog-HDL and VHDL (Very High Speed Integrated Circuit Hardware Description Language) , is used for writing a particular version of processor. This helps the designer to use them in any of the embedded applications. These can be used in the processor just by embedding a particular application in the processor. RISC (Reduced Instruction Set Computer) is an efficient Computer Architecture which can be used for the Low power and high speed applications of the processor. RISC Processors are important in application of pipelining. The curb of the processor is the Instruction Set Architecture used for developing it. The total worthiness of the processor depends on utilizing the Instruction Set Architecture.

Instruction Set Architecture is a metaphysical interface between Low level system of the machine and the hardware, that contain all the information about the machine , required to write a program for the machine .Instruction Set Architecture can shortly be defined as the architecture which defines working of the instruction's in the processor. It also defines how a processor takes inputs and the size of the processor. A large number of instruction sets are available in the market. Examples of Microprocessor's based on ISA's are SPARC, Hitachi, Power PC, Motorola68k, IA32 and ARM.

However a lot of research is being carried out in the field of processor's to satisfy the performance issues. But now a days it is mandatory to use a machine which is efficient in the terms of speed, power, performance and size. Though there are tradeoffs between all the performance parameter's, Research is being carried out to satisfy all the above performance parameters.

Though a more number of instruction sets are available in the market, an instruction set which can handle less power, high speed; low area needs to be selected for the efficient functioning. To satisfy all the above requirements we consider the MIPS (Microprocessor without Interlocked Pipelining Stages) Instruction Set Architecture [1].

This paper totally is a zoom in on how the MIPS instruction Set is embedded in an RISC Processor. This paper also concentrates on reducing the power utilized by the processor in order to satisfy the Low Power constraint of the developed Processor.

II.BACKGROUND OF MIPS

MIPS can be abbreviated as Microprocessor without Interlocked Pipelining Stages. It was first developed by Sony, Nintendo and NEC. This was developed to overcome the problems of the conventional design i.e. using same instruction set for all the applications makes instruction set busy and system a delaying system. Hence MIPS is made as an alternative to conventional RISC Processor.

A. Instruction Set of MIPS[2]

The Instruction Set of MIPS is divided for three different type of instructions separately.

1. Register Type (R-Type)

2. Immediate Type (I-Type)

3. Jump Type (J-Type)

1. Register Type:

This instruction type is used for all the arithmetic operations of the circuit. The instruction set representation of R-Type is shown below:

31-26	25-21	20-16	15-11	10-6	5-0
OP-Code	Reg_s	Reg_t	Reg_s	Shift Amount	Function

OP-Code defines the operational code which notifies other unit to perform its work.

Reg_s is the source register for which computation is carried out. Reg_t is an another source register used for computation. Reg _d is a Destination Register used for storing the address where the instruction needs to be stored. Function block notifies the ALU which instruction to be executed. Instructions used by this type of Instruction Set are addition, subtraction and other arithmetic calculations.

2. Immediate Type:

This instruction type is used for all the immediate addressing modes of MIPS. All the logical calculations related to memory are performed here.

31-26	25-21	20-16	15-0
OpCode	Reg_s	Reg_t	Immediate Address

The instruction Set representation of this instruction is shown above. Immediate Address gives the offset address added to the regular address.

3. Jump Type:

This instruction type is used for all the instruction which use jump instructions. Only the Jump type instructions are used here. The instruction set of this type of instruction is

31-26	25-0
OpCode	Target Address

shown here. Target Address is the address where the instruction which is jumped to be saved.

B. Structures used in MIPS Design

The MIPS processor is compilation of a lot many structures which are used as blocks of interconnection. The utilization of all these blocks is explained here.

1. Instruction Memory: This can simply be replaced by a

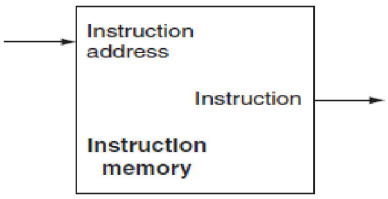


Figure 1 : Instruction Memory

Re ress is the addres y. For the particular address respective instruction is saved in the ROM. It just reads the particular instructions from the required address. Hence this part in this work is replaced by an ROM.

2. *Instruction Decoder Unit:* This unit generally takes input as a 32-bit instruction and outputs all the required instructions and these specified values are input to next part of circuit. The block diagram of Instruction Decoder unit is shown here.

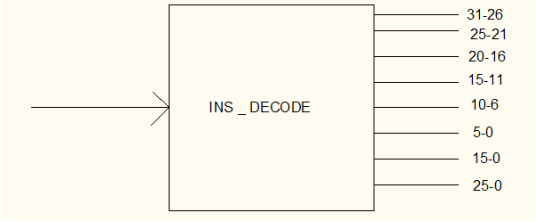


Figure 2: Instruction Decoder

3. Data Memory: This is the second memory of the MIPS. This memory can be used for both Read and Write applications. Hence this circuit can be replaced by a Random Access Memory. The input is the address and if read signal is enable it reads the data from the inputs memory location. If write signal is enable it writes the data into the input's memory location.

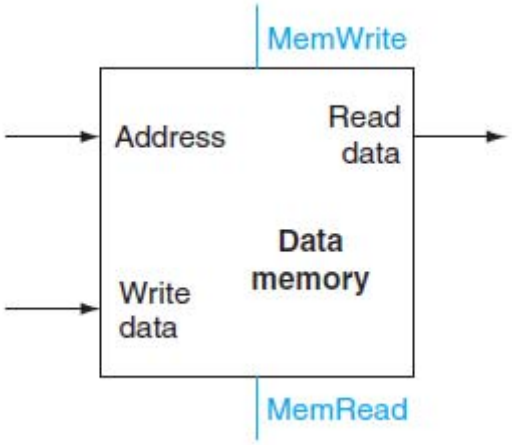


Figure 3: Data Memory

4.Sign Extend Unit : This unit is sign extension unit which inputs an 16 bit instruction as input and which gives 32 bit instruction as output . It is used for extending the Immediate value.

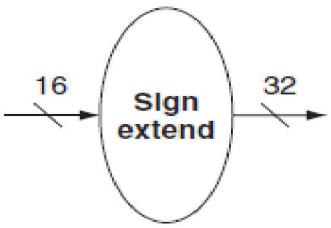


Figure 4: Sign Extend Unit

5. Registers : This is the next block presiding the Instruction Decoder. It is useful for reading the Instruction and performing the reading or writing operations on the particular

memorylocation.

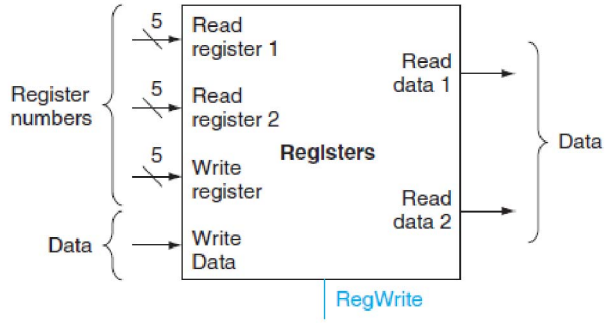


Figure 5: Registers

6. *Datapath :* Datapath can be explained as the interconnections used for connecting the blocks. When connecting the blocks we can have two types of implementations. There are types of datapath's.

a) Single Datapath : This type of data path is an simple data path. It is called single as it makes an endeavour to complete Instruction in one clock cycle . The disadvantage of single datapath is when an Instruction is not able to complete in a clock cycle , it stops immediately giving erroneous result. To overcome the problem we move on to multi core datapath.

b) Multicore Datapath : It is an instant when designer divides the execution of Instructions into different clock cycles, Inorder to ensure that even the slowest one of all the instructions execute. Multicore is an efficient way of implementaion for its questerity.

B. Addition of Register's to Instruction Set

1. Adding Registers to R-Type Instruction[2] This type of operations does not need any memory element, just they take the input and perform the required calculation and gives the output .The output of ALU is written back to the write register of Register Block.

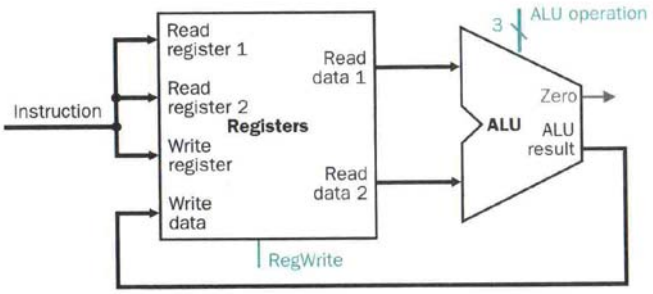
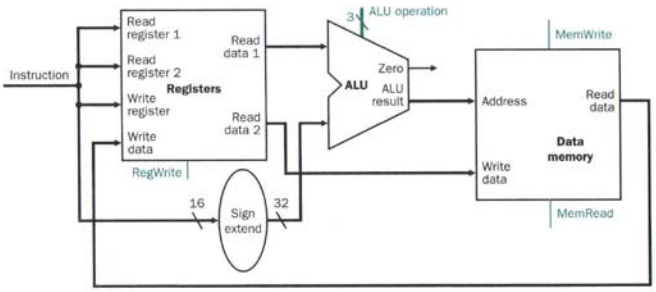


Figure 6: Adding Registers to R-Type Instruction

2. Adding Registers to I-Type Instruction [2]: These types of operations are used for reading and writing from the memory. This type of instruction depends only on memory. The load word and store word are the examples of I - Type of instruction. Hence we use RAM for memory operations.

Figure 7: Adding Registers to I-Type Instruction



3. Adding Registers to J-Type Instruction [2]: These type of operations are used just for jumping the memory location and outputs the required jump address i.e. address location where the value needs to be copied

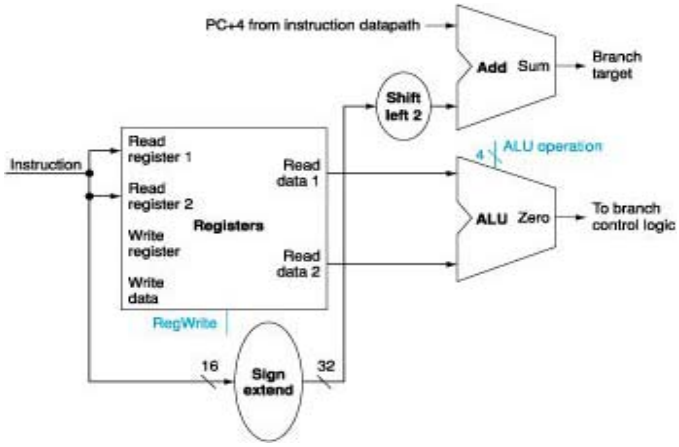
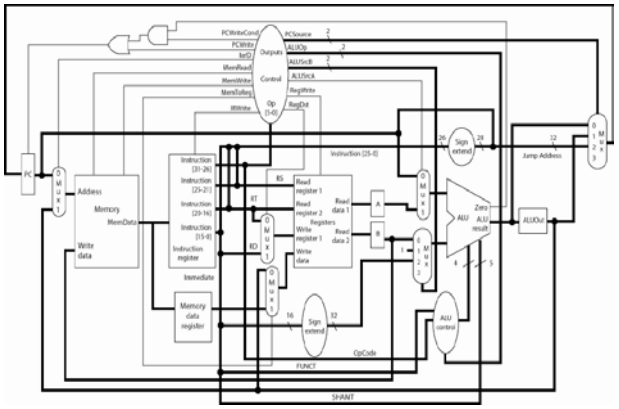


Figure 8: Adding Registers to J-Type instruction.

C. Compilation of all the Blocks

Hence by combining all the blocks together into one total block diagram gives MIPS Multicore Data path[3][2].

Figure 9: Block Diagram of MIPS Multi core Data path



This block diagram is used for multi cycle implementation of MIPS. When sequence of instructions are input, then circuit needs to determine which instruction to be executed. To overcome this Problem we add Pipeline technique to the above block diagram.

D. *Adding Pipeline to the Stages[3]*

To add Pipeline to the block, it needs to be divided into parts based on our requirement. These blocks are used for

moving the address and helps in the execution. The separate blocks used for pipelining are

1. Instruction Fetch
2. Instruction Decode
3. Instruction Execute
4. Memory Access
5. Write Back

1. Instruction Fetch [4]: This block of pipelining is used for storing the address of next instruction and to fetch the instruction in the respective address location. Pcsrc is the select line which we need to select the program counter.

2. Instruction Decode [4]: This is the decoding stage used for decoding a particular instruction into parts and sending the same instruction for the execution. Reg_Write is the signal used for enabling the output to be written on write register.

3. Instruction Execute: The heart stage of processor used for executing the given inputs. The ALU and ALUControl play an active operation in this mode. The Reg_DST is the signal used for selecting the register destination to store the value. ALU source is the signal used for selecting which ALU source the value has to be taken for comparison. ALUop is the signal used for selecting the ALU operation.

4. Memory Access Stage: This stage is used for accessing the memory locations of the processor. The basic block of this stage is RAM unit. Mem_read is the signal used for enabling the RAM to read it from the memory location. Mem_write is the signal used for enabling the RAM to write to the memory location.

5. Write Back Stage: This stage is used to select whether to write back to decode stage. Mem_reg is the signal used to decide whether to write back to memory.

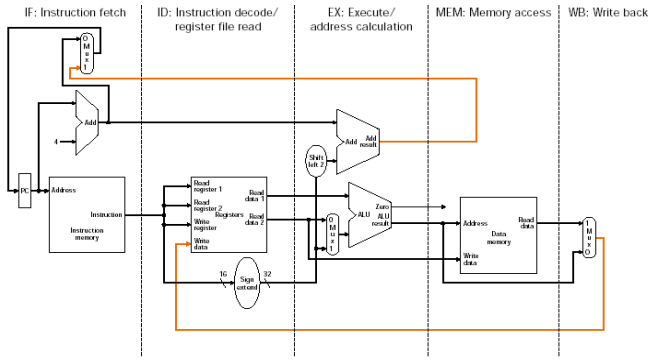


Figure 10 :Addition of Pipeline to MIPS Datapath

The disadvantage of direct Pipeline is memory will not be ready for the instruction which creates an additional delay for execution. The basic example for the above problem is considering load word and store word instructions in a row then memory will not be ready until instruction load word completes. To overcome this problem we need to add interstage buffers which diminish the delay by larger extent.

D. Adding the interstage buffers

When the interstage buffers are added the data from one block is input from one buffer and output to another buffer. When the system needs particular input it takes the particular input from the respective interstage buffer [2]. The Disadvantage of the interstage buffer Pipeline is if the instruction is to be sent as input to write back stage then system needs to travel all along the system, which steers an additional wiring delay.

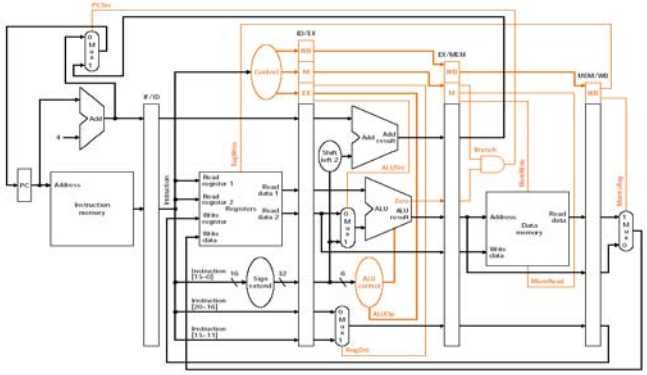


Fig 11: Pipelined MIPS With Interstage Buffers

Hence the respective buffers to particular inputs are taken and if the input is to be given to the particular block it is given to the buffer. It greatly enhances the performance. This is an exemplary circuit for pipelining.

III.PROBLEMS OF MIPS MULTI CORE DATAPATH

When the next instruction is prevented from being executed in the respective clock cycle is called a Hazard [3]. The same on MIPS Data path was reckoned as a hazard. Corresponding to MIPS multi core Data path there are three types of hazards.

1. Structural Hazards: When two or more instructions strive to utilize a single resource at the same moment, it results in Structural Hazard. In accordance to corresponding instruction if it needs additional hardware than available then Structural Hazard arises [2]. The occurrence of these hazards is meagre. As an example when single memory is utilized as RAM and ROM i.e. Instruction Memory and Data Memory then Structural Hazard may occur. The solution to master the problem is by delaying the later instruction accessing the memory by one clock cycle or provide separate memory elements for instruction memory and data memory.

2. Data Hazards: If the respective instruction needs data from previous instruction then the data hazard arises [2].The dependency exists between two variables if one variable needs to read the result of other variable. It is also called as Read-after Write Hazard. The instruction completes its write operation in stage 5 and when others instruction needs to , access it in stage 2 then the hazard occurs. This is also called as Data Dependence Hazard.

Considering the following instruction's

Add $9, $15, $20
Sub $12, $9, $15

And $21, $13, $9

Here no dependency problem exists with $15 as it is just used as a source register. The major problem exists with $9 as it is used as destination register in "add" instruction and source register in "sub" instruction. The instruction "Add" needs particular clock cycle and before the completion of clock cycle the "sub" instruction tries to access the data which creates a hazard[8].

There are many ways to resolve a data hazard. One way is by utilizing a Hazard Detection Unit. This stalls a pipeline on the incidence of hazard. This is done by using a "nop" instruction. Stalling can implicitly be defined as slotting a "nop" instruction into stages. It increases the delay but we can reduce the data hazard by stalling a particular pipeline [5]. Another way of resolving the data hazard is by forwarding the result i.e. sending the result in advance. Usage of a forwarding unit is an efficient technique. The required data is being predicted and required steps are taken to control the hazard by bypassing the data in advance. If the predicted data is found to be wrong then total output will be in vain. Another method is by flushing. Prophesing a branch prior execution is called flushing. If branch forecasting is found to be false then we need to change the coding back again [5].

3. Control Hazards: When we need to find the destination of the branch or branch address but we are unable to fetch any new instruction until we know the destination If they occur there will be change in respective program flow, which creates a control hazard [2]. Examples of these are branch and jump instructions changes the program counter creating the problem of control hazard. These control hazards can be shrinked in many ways. One of the techniques is code-reordering. Rearranging the code for the efficient use is called ordering this work needs to be done before execution. Another method is branching. Branch is predicted earlier and if it is found to be false then we need to flush an existing instruction[5].

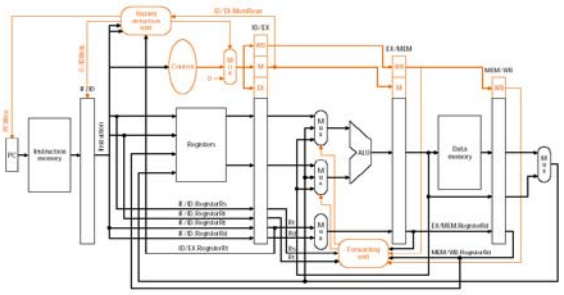


Figure 12: Pipelined MIPS using Hardware Detection Unit

Hence Data forwarding unit is used in the execution stage to predict the values in advance [5]. An efficient Hardware detection unit is used to detect the hardware errors and find the corresponding solution for the respective units.
The hazard detection units are integrated into the processor making it different from the previous designs. Hence our processor is combination of all the things including the Hazard Detection Units.

The aim to design this hazard detection unit is to find out the hazards in advance and rectify them before they occur. So consecutive steps are taken to prevent these hazards and remove them. The best ways to fix all of them are by stalling the processor when the hazard occurs. But this is not an ideal way of implementation. To overcome the problem we use the forwarding mechanism i.e. bypassing the inputs just by using the multiplexer's and just knowing them in advance. This can be done by using multiplexors at the input of the ALU, to know which inputs are to be given to ALU. This can be done by taking the execution memory Read register is not equal to zero and also when the execution memory write register is equal to execution memory read register as well when the memory write back read register is not equal to zero as well as when memory write back read register is equal to write register. This needs to be controlled by using the proper hardware. Thus output of ALU is forwarded. In doing so the complexity arises during the memory write back register used as a forwarding unit. To overcome that problem we use the hazard detection unit to complete processor. The Hazard Detection Unit is given as input to every stage where there is a change of occurrence of hazard. The hazard detection unit is generally used for stalling a pipeline if any hazard is found in advance. In the Instruction Decode Stage, Hazard Detection Units are used to check if the source address is similar to destination address and if so it stalls the pipeline in advance. This stalling can be done on hardware by making the program counter register to change.

IV.ADDITION OF LOW POWER METHODOLOGY

Low Power Methodologies are the methods used to reduce the Power utilization of the circuit. These techniques are really helpful in reducing the total power used by the circuit. Out of all these methods a few of them are used in our paper.

E. Clock Gating

The total power availed oneself is the tally of static and dynamic power. Dynamic Power can be reckoned be the formula $C_L V_{DD}^2 F$. Here the C_L is the load capacitance utilized by the circuit., Vdd is the Destination [6]. F is the frequency of the signal. here the focul idea is to shrink clock circuit as clock utilized almost 14-45% of total power . Hence the focal area is to shrink clock circuit partitioning by diminution of unnecessary Switching , Switched Capacitance , and by Lowering the Clock Circuitry.

F. Multi V_t

The multi V_t can be abbreviated as Multi Threshold. The optimization of MVT does not change the placement of the circuit. Swapping of the cells is available during the optimization [7].

G. Reduction of Dynamic Leakage

The dynamic power is power consumed when switching from one state to another. It can also be called as Short circuit power. The amount of power wasted is called as leakage Power. Reducing this power enhances the functioning of the circuit[8].

v.Results and analysis

. Table 1: Comparisons of Parameters

Technology	Baseline (generic)	180nm Technology	Clock gating At 180nm	Clock gating + Multi V_t
Leakage Power(µw)	1.1	1.463	2.317	1.904
Total power(µw)	318.4	30.286	5.824	4.611
Area (No. of cells)	558	560	760	725
Frequency (Ghz)	1.2	1.16	1.16	1.16

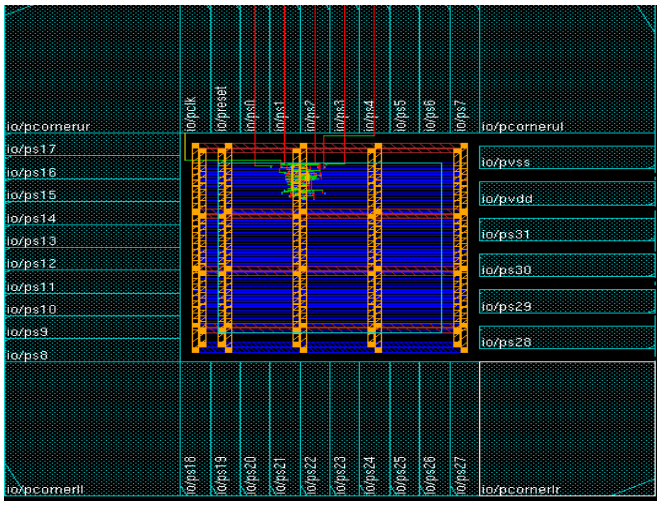


Figure 13: Layout with IO pads

The architecture was designed and simulated in verilog-HDL and Nc-launch of Cadence respectively. The designed architecture was synthesized in RTL compiler at 180nm technology in different configurations as shown in table 1. The physical design part of architecture was carried on by using SOC-encounter tool. The layout was successfully verified using DRC and LVS. Figure 13 shows the layout design with IO pads. Figure 14 shows the amoeba view of layout which displays the density of cells on chip. All tools used in this project are licensed by Cadence Inc.

From the results tabulated in table1, we can observe the power is been reduced by 90% when compared to total power in generic consequently almost 25% more leakage power was experienced. At lower technology nodes the threshold voltage is less therefore leakage power is more, hence we are experiencing a significant increase in leakage power. Clock gating and Multi V_t results shows still reduction in total power dissipation but at the cost of area. Variable threshold voltage decreases the leakage current consequently leakage power also decreases. Almost 27% more area occupation is observed when compared with mapped technology without clock gating. The analysis was performed by maintaining the same frequency inorder to have same platform.

Figure 14: Density of cells on chip

The details of the final Column are retrieved by applying Low power technique's for the processor with hazard detection unit integrated in it. Hence by justification to the name the project takes very low power.

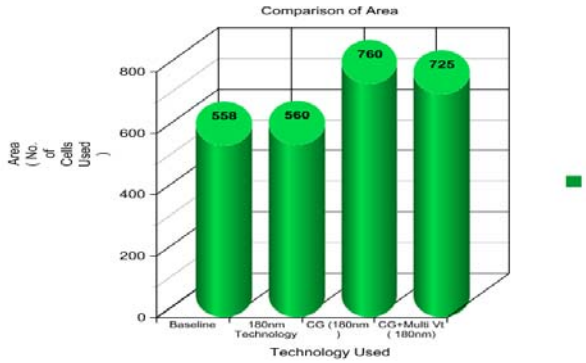


Figure 15 Graph showing Comparison in Area

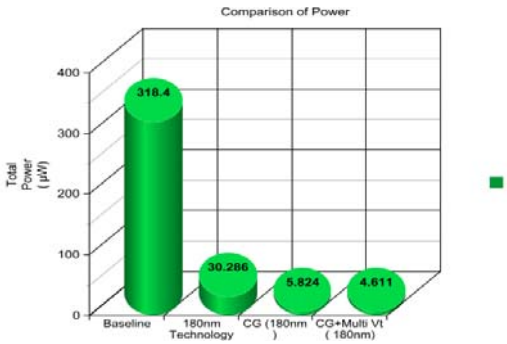


Figure 16 Graph Showing Comparison in Power

VI. CONCLUSION

There by we can observe that the Hazard Detection Unit by name detects the hazards and tries to reduce them to get an efficient output. Hence the Hazard Detection Unit helps for accurate behaviour of the system. Moving Hazard Detection Unit on to the processor reduces the delay. Application of low power methodology to MIPS processor reduces 90% of the total power and makes it more efficient. Though there is 27% increase in area the efficiency of the processor in terms of power makes it an ideal system. Cadence tool and Xilinx were used to verify the output. The project is advanced by the utilization of Hazard Detection Unit , Uses low power technique and MIPS instruction Set. Hence the justification is done to the name of the project. This work can be enhanced by adding an Built In Self Test (BIST) as a testing mechanism which ensures the correct functionality of the Processor. The instruction set can still be increased by increasing number of instructions which makes the system more complex. The Data gating technique can also be applied to the work to check the efficiency of the processor. Comparisons can be made between different Low power techniques to make it further more efficient.

VII.Acknowledgment

I thank Prof C.Prayline Rajabai for her help in making the paper look awe-inspiring. I whole heartedly thank all the faculty of the department for their indebted help in doing this paper.

References

[1] Gautham.P, ParthasarathyR, Karthi Balasubramanian, “Low-Power Pipelined MIPS Processor Design”, ISIC 2009.Pg :462-465

[2] David A Patterson, John L Hennessy "Computer Organization and Design - The Hardware-Software Interface" 3rd Edition

[3] Munmun Ghosal “ A VLSI Design approach for RISC based MIPS architecture “ IJAEE Volume 2 pb 2012 Pg 45-49

[4] Rupali S. Balpande , Rashmi S. Keote, " Design of FPGA based Instruction Fetch and Decode Module of 32 bit RISC (MIPS) Processor 2011" ICCSNT Pp. 409-413

[5] Zulkifli , Yudhanto , Soetharyo " Reduced Stall MIPS Architecture using Pre-fetching Accelerator " ICEEI 2009 Pg 611-616

[6] Linder , Schimd " Processor Implementaion in VHDL" University of Ulster 2012 pg:1-20

[7] Charles Brej.,“A MIPS R3000 microprocessor on an FPGA”, 13 February 2002

[8] I.Dalal, A.Ganesh, Aishwarya.D “An 8 bit Power-Efficient MIPS Processor “ Advanced VLSI Design Sping 2014 Pg: 1-6