FPGA IMPLEMENTATION OF LOW POWER PIPELINED 32-BIT RISC PROCESSOR

Preetam Bhosle, Hari Krishna Moorthy

Abstract: This paper presents the design and implementation of a low power pipelined 32-bit High performance RISC Core. The various blocks include the Fetch, Decode, Execute and Memory Read / Write Back to implement 4 stage pipelining. In this paper we are proposing low power design technique in front end process. Harvard architecture is used which has distinct program memory space and data memory space. Low power consumption helps to reduce the heat dissipation, lengthen battery life and increase device reliability. To minimize the power of RISC Core, clock gating technique is used in the architectural level which is an efficient low power technique. 7-SEG LEDs are connected to the RISC IO interface for testing purpose, Verilog code is simulated using Modelsim and then implementation is done using Altera Quartus II and Altera FPGA board.

Index Terms – Architectural level power reduction, Auto branch prediction, Clock Gating, High performance architecture.

I. INTRODUCTION

Low power has emerged as a principle theme in today's electronics industry. The need for low power has caused a major paradigm shift where power dissipation has become an important consideration as performance and area. RISC is termed as *R*educed *I*nstruction *S*et *C*omputer, computer arithmetic-logic unit that uses a minimal instruction set, emphasizing the instructions used most often and optimizing them for the fastest possible execution. This processor will follow the RISC architecture because it supports a predefined set of instructions. In this all the instructions have same length. Software for RISC processors must handle more operations than traditional CISC [*C*omplex *I*nstruction *S*et *C*omputer] processors, but RISC processors have advantages in applications that benefit from faster instruction execution. They are also less costly to design, test, and manufacture.

MIPS processor design is based on the RISC design principle that emphasizes on load/store architecture. Due to the difference in time taken to access a register as compared to a memory location, it is much faster to perform operations in on chip registers rather than in memory. To eliminate the latency of memory operations, MIPS processor uses the load/store architecture where the access to memory is only through load and store instructions. The processor has many registers and operations are performed in data present in those registers.

The basic architecture of a MIPS processor is shown in Fig. 1

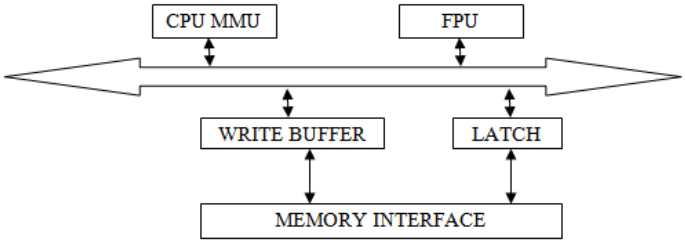

Figure: 1 MIPS Top Level Structure

When talking about the characteristics of MIPS processors, it is very important to differentiate between the terms 'architecture' and 'hardware implementation'. Architecture refers to the instruction set, registers, address, layouts etc. while hardware implementation refers to the manner in which different processors use the architecture to build their own model. The architecture remains the same for all MIPS based processors while the implementations may differ.

The pipelined architecture of the processor is given in Fig. 2, where the different stages of the pipeline are separated by pipeline registers. The major building blocks of the design include

- Memory and Register Blocks
- Datapath
- Control Logic
- Low Power Unit.

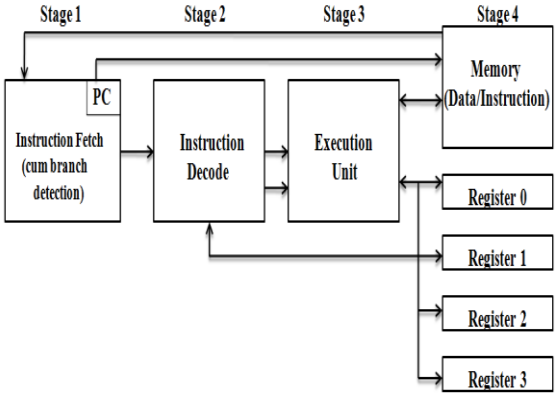

Figure: 2 Different Blocks of Pipelined Architecture

Manuscript received May 01, 2012. (Fill up the Details)

Preetam Bhosle, Department of Electronics and Communication Engineering, , School of Engineering and Technology, Jain University Bangalore, India. (E-mail: pritam044@gamil.com).

Asst. Prof. Hari Krishna Moorthy, Department of Electronics and Communication Engineering, School of Engineering and Technology, Jain University, Bangalore, India. (E-mail: locushari@gmail.com).

Memory and Register Blocks

The data and instruction memories are 32X 256 bytes wide and 32X1024 bytes wide respectively while the register block is of size 32 bytes. The instruction memory is implemented as a single port on-chip distributed ROM while the data memory is implemented as a single port on-chip block RAM inside the FPGA.

Datapath

The pipeline stages for different type of instructions are processed as follows, in the fetch stage, instructions are fetched at every cycle from the instruction memory whose address is pointed by the program counter (PC). During the decode stage, the registers are read from the register file and the opcode is passed to the control unit which asserts the required control signals. Sign extension is also done for the calculation of effective address. In the execute stage, for Register type instructions, the ALU operations are performed according to the ALU operation control signals and for load and store instructions, effective address calculation is done. The load and store instructions write to and read from the data memory in the memory stage while the ALU results and the data read from the data memory are written in to the register file by the register type and load instructions respectively in the write-back stage.

Control Logic

The pipeline is controlled by setting control values during each pipeline stage. Each control signal is active only during a single pipeline stage and hence the control lines can be divided according to the four pipelined stages. These signals will be forwarded to the adjacent stage through the pipeline registers.

Data forwarding cannot prevent all pipeline stalls. When a data requested by a load instruction has not yet become available it leads to load-use hazards. To resolve this hazard, the pipeline is stalled by the unit for one stage and then continues with the forwarding of data.

Low Power Unit

The main power reducing technique that has been explored in this architecture is clock gating. Clock gating is a method where the clock signal is prevented from reaching the various modules of the processor. The absence of the clock signal prevents any register and/or flip-flop from changing their value. As a result of this, the input to any combinational logic circuit remains unchanged, and thus no switching activity takes place in those circuits. Since, in CMOS circuits, most of the power dissipation results from switching activity, clock gating greatly reduces the overall power consumption.

II. PROPOSED SYSTEM

The RISC processor was designed using pipelined architecture; through this the speed of the operation as well as overall performance was increased. In this 4-stage pipelining is implemented.

The 4 stages of pipeline are

1. Fetch
2. Decode
3. Execute
4. Memory Read / Write Back.

During the design process, low power technique was included in the architectural level, also this methods is expected to be efficient than back-end low power reduction techniques. For increasing the performance, latch based clock gating technique is used. Clock gating is the most popular method for power reduction of clock signals. When the clock signal of a functional module is not required for some extended period, we using a gating function to turn off the clock feeding the module. Clock gating saves power by reducing unnecessary clock activities inside the gated module. The design complexity and performance degradation of clock gating are generally manageable.

Processors with pipelining consist internally of stages (modules) which can semi-independently work on separate microinstructions. Each stage is linked by flip flops to the next stage (like a "chain") so that the stage's output is an input to another stage until the job of processing instructions is done. Such organization of processor internal modules reduces the instruction's overall processing time. An instruction pipeline is said to be fully pipelined if it can accept a new instruction every clock cycle. A pipeline that is not fully pipelined has wait cycles that delay the progress of the pipeline.

The proposed architecture is general-purpose RISC processor with pipelining feature; it gets instructions on a regular basis using dedicated buses to its memory, executes all its native instructions in stages with pipelining. It can communicate with external devices with its dedicated parallel IO interface.

There are basically three types of instructions namely Arithmetic and Logical Instructions (ALU instructions), Load/Store instructions and Branch instructions.

1. ALU Instructions:

Arithmetic operations either take two registers as operands or take one register and a sign extended immediate value as an operand. Some arithmetic instructions are ADD, SUB, and MUL. The result is stored in a third register. Logical operations such as AND OR, XOR do not usually differentiate between 32-bit and 64-bit. Other logical instructions are NAND, NOR, NOT etc.

2. Load/Store Instructions:

Usually take a register (base register) as an operand and a 16-bit immediate value. The sum of the two will create the effective address. A second register acts as a source in the case of a load operation. In the case of a store operation the second register contains the data to be stored. Examples are: LW, SW etc.

3. Branches and Jumps Conditional branches are transfer of control. A branch causes an immediate value to be added to the current program counter. Some common branch instructions are BZ (Branch Zero), BRZ (Branch Register Zero), JMP (Jump Instruction), JMPZ (Jump when zero) etc.

III. ARCHITECTURE

The proposed architecture consists of 4-stages namely:

- Instruction Fetch (IF),
- Instruction Decode (ID),
- Execution unit (EX),
- Memory Unit (MU),
- Memory write/read unit
- Low power unit along with 4 general purpose

registers namely Register0, Register1, Register2 and Register3.

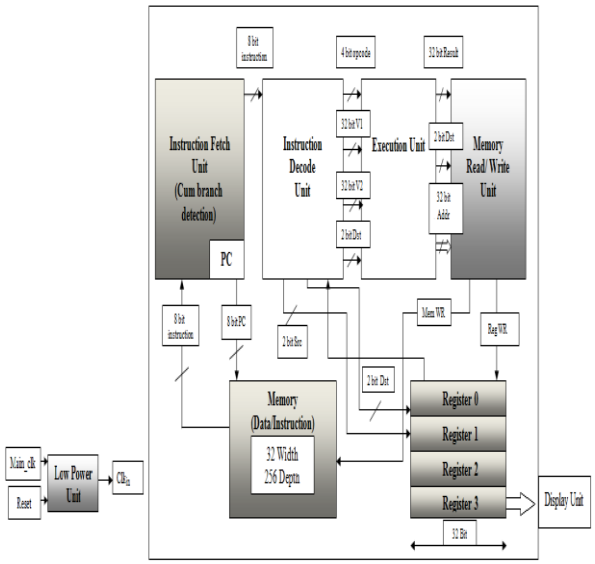

Fig. 3 Proposed Architecture of Low power RISC Processor.

Instruction Fetch (IF):

The Instruction Fetch stage is where a program counter (8 bit) will pull the next instruction from the correct location in program memory. In addition the program counter will be updated with either the next instruction location sequentially, or the instruction location as determined by a branch.

Instruction Decode (ID):

The Instruction Decode stage is where the control unit determines what values the control lines must be set to depending on the instruction. In addition, hazard detection is implemented in this stage, and all necessary values are fetched from the register banks.

Execute Unit (EX):

The Execute stage is where the instruction is actually sent to the ALU and executed. If necessary, branch locations are calculated in this stage as well. Additionally, this is the stage where the forwarding unit will determine whether the output of the ALU or the memory unit should be forwarded to the ALU's inputs.

Memory Unit (MU):

Finally, the Memory Access stage is where, if necessary, system memory is accessed for data. Also, if a write to data memory is required by the instruction it is done in this stage. In order to avoid additional complications it is assumed that a single read or write is accomplished within a single CPU clock cycle.

Short Instruction

Opcode				Source		Destination	
0	0	1	0	0	1	1	0

Long Instruction

Opcode				Source		Destination	
0	1	1	0	1	0	?	?
Address							
0	0	0	1	1	1	0	1

Table 1: Instruction set for the proposed Architecture

Instruction	Instruction Word			Action
	Opcode	Source	Destination	
NOP	0000	??	??	None
ADD	0001	Src	Dst	Dst <= Src + Dst
SUB	0010	Src	Dst	Dst <= Src - Dst
AND	0011	Src	Dst	Dst <= Src & Dst
NOT	0100	Src	Dst	Dst <= ~ Src
NAND	1001	Src	Dst	Dst <= ~ (Src & Dst)
MUL	1010	Src	Dst	Dst <= (Src[15:0] * Dst[15:0])
RD*	0101	??	Dst	Dst <= Memory [ADD_R]
WR*	0110	Src	??	Memory [ADD_R] <= Src
BR*	0111	??	??	PC <= Memory [ADD_R]
BRZ*	1000	??	??	PC <= Memory [ADD_R]
HALT	1111	??	??	Halts execution until reset

Note: Where * denotes the instruction requires second word of data, and?? denotes don't care.

The instruction set used in this architecture consists of arithmetic instructions, logical instructions, memory instructions and branch instructions. It will have short (8-bit) and long (16-bit) instructions. For all Arithmetic and logical operations 8-bit instructions are used, and for all memory transactions and jump instructions 16-bit instructions are used. It will also have special instructions to access external ports.

Operation of Instructions:

1. Arithmetic ALU operation (3)

ADD: Dst = Src + Dst
Operands var1 and var2 stored in register locations Src and Dst are added and written to the destination register specified by Alu_out.
SUB: Dst = Src - Dst
Operand var2 is subtracted from Operand var1 and is written to Alu_out.
MUL: Dst <= (Src[15:0] * Dst[15:0])
Src and Dst are multiplied and is written to Alu_out

2. Logical ALU operation (4)

AND: Dst = Src & Dst
Operand var1 is bitwise anded with Operand var2 and written into Alu_out..

NOT: Dst = ~Src
Operand var1 is bitwise inverted and written into Alu_out.

NAND: Dst = ~ (Src & Dst)
Operand var1 and var2 first anded and then inverted and written to Alu_out.

3. Memory operations (2)

RD*: Dst = Mem[ADD_R]
The memory word specified by the memory location of ADD_R is loaded into register Dst.

SW*: Mem[Rs] = Rt
The data in register Dst is stored into the memory location specified by ADD_R.
Both the instructions require second word of data.

4. Conditional Branch operations (2)

BZ*: PC = Memory [ADD_R] if Src=0
If all the bits in register Src are zero than the current Program Count (PC + 1) is offset to PC + 1 + Offset. The count is offset from PC+ 1 because it is incremented and stored during the Fetch cycle.

BNZ*: PC = Memory [ADD_R] if Src! = 0
If all the bits in register Src are not zero than the current Program Count (PC + 1) is offset to PC + 1 + Offset.
Both the instructions require second word of data.

Addressing Modes Used:
In this proposed architecture different addressing modes have been used namely:
Direct addressing modes,
Immediate addressing mode,
Register mode and PC relative mode.

Features of Proposed Architecture:
- 4 - Stage 32-bit processor.
- Provides hazard detection unit to determine when stall must be added.
- Provides halt support.
- High performance.
- Pipelining would not flush when branch instruction occurs as it is implemented using dynamic branch prediction.
- Take care of order-of-execution.

IV. SIMULATION RESULTS

Modelsim is used for simulation and results have been verified. Once the functional simulation is done, it is implemented using QUARTUS-II and Altera FPGA board. Testing & debugging of the project was done on FPGA development board. The complete project has been developed on FPGA Development board platform, which has different testing & debugging scopes such as on-board push-button switches, toggle switches and LEDs. So, all the testing has been carried out on the board. 7-SEG LEDs is connected to the RISC IO interface for testing purpose. Different patterns have been generated with RISC instructions, and are sent to 7-SEG LEDs through parallel IO interface of RISC.

Here in this ALTERA family, many different devices were available in the tool. In order to implement this design the device named as “QUARTUS-II” has been chosen and the package as “EPF20F484C7”is used on ALTERA DE1 board platform. The design of 32bit-RISC is simulated and its results were analyzed as follows.

The RTL schematic of the proposed architecture is as shown in Fig.4

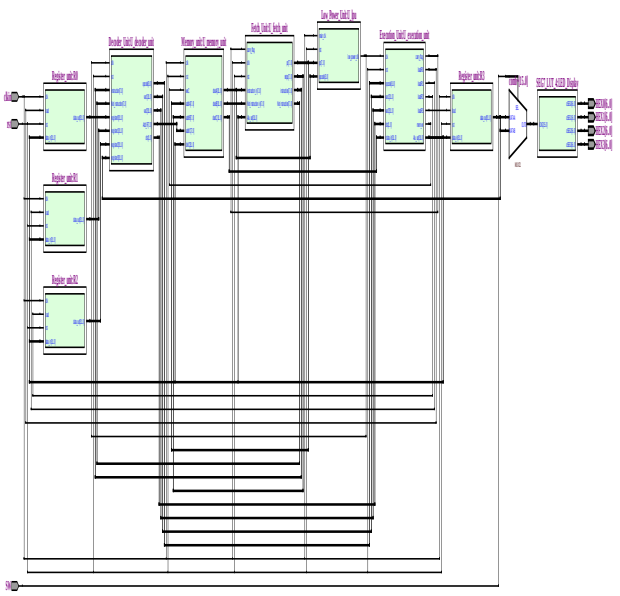

Figure: 4 RTL Schematic

Design Summary, Simulation and Synthesis Result:
Design Summary
The device utilization summary is shown below in which it gives the details of number of devices used from the available devices and also represented in %. Hence as the result of the synthesis process, the device utilization in the used device and package is shown below.
In timing summary, details regarding time period and frequency shown are approximate when synthesized. After place and route is finished, we get the exact timing summary.

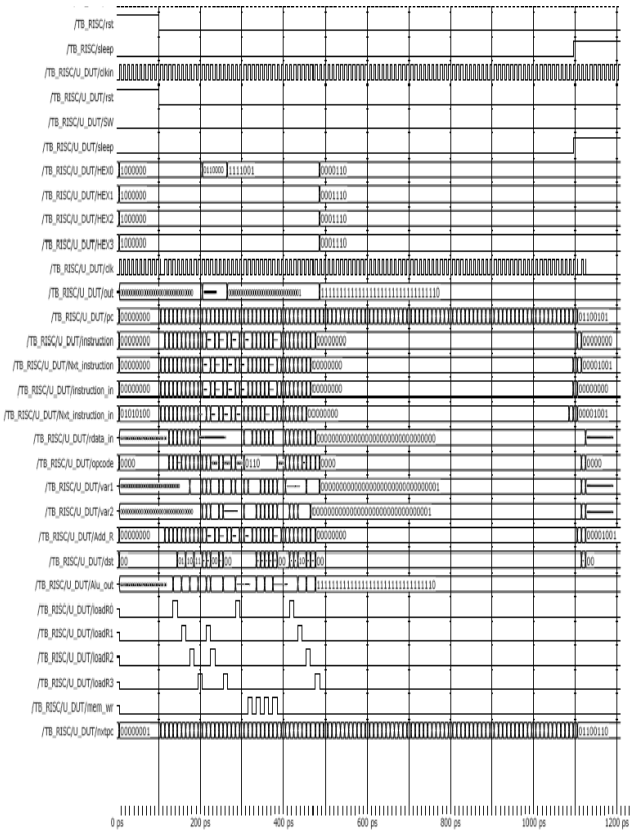

Figure: 5 Simulation Waveform

Flow Summary	
Flow Status	Successful - Thu Jul 19 10:47:42 2012
Quartus II 32-bit Version	12.0 Build 178 05/31/2012 SJ Web Edition
Revision Name	RISC_SPMLP
Top-level Entity Name	RISC_SPMLP
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	14,131 / 18,752 (75 %)
Total combinational functions	8,973 / 18,752 (48 %)
Dedicated logic registers	8,479 / 18,752 (45 %)
Total registers	8479
Total pins	32 / 315 (10 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	2 / 52 (4 %)
Total PLLs	0 / 4 (0 %)

Figure: 6 Flow Summary

Analysis & Synthesis Summary	
Analysis & Synthesis Status	Successful - Thu Jul 19 10:41:59 2012
Quartus II 32-bit Version	12.0 Build 178 05/31/2012 SJ Web Edition
Revision Name	RISC_SPMLP
Top-level Entity Name	RISC_SPMLP
Family	Cyclone II
Total logic elements	17,251
Total combinational functions	8,941
Dedicated logic registers	8,479
Total registers	8479
Total pins	32
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	2
Total PLLs	0

Figure: 7 Analysis and Synthesis Summary

PowerPlay Power Analyzer Summary	
PowerPlay Power Analyzer Status	Successful - Thu Jul 19 10:48:05 2012
Quartus II 32-bit Version	12.0 Build 178 05/31/2012 SJ Web Edition
Revision Name	RISC_SPMLP
Top-level Entity Name	RISC_SPMLP
Family	Cyclone II
Device	EP2C20F484C7
Power Models	Final
Total Thermal Power Dissipation	2549.73 mW
Core Dynamic Thermal Power Dissipation	2373.28 mW
Core Static Thermal Power Dissipation	53.20 mW
I/O Thermal Power Dissipation	123.25 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Figure: 8 Power Analysis Summary

V. CONCLUSION

The design of 4-stage 32-bit low power RISC processor performing arithmetic, logical, memory and branch instructions is presented in this paper. An architecture is devised in order to felicitate the writing of codes in Verilog. The Verilog coding synthesis issues play a vital role in the speed-area optimality because RTL schematic depends heavily on how we have coded in Verilog. The design is simulated on ModelSim SE 6.4e tool and then synthesized and verified on Altera FPGA board. The proposed architecture is able to prevent pipelining from flushing when branch instruction occurs and able to provide halt support. It can be inferred from the synthesis report that proposed architecture offers speed which is approximately 110.92 MHz's.

ACKNOWLEDGMENT

This paper is dedicated to our parents, and family for their love, endless support and encouragement. We would also thank our friends, Mr. B. Ahmed and Mr. Sunil for their constant support and guidance We would like to thank the Department of Electronics and Communication Engineering, management of School of Engineering and Technology, Jain University, Bangalore for their constant support and encouragement in undertaking the work.

REFERENCES

1. M.E. Hopkins, "A Perspective on the 801/Reduced Instruction Set Computer", *IBM Systems Journal*, Vol. 26, No. 1, pp. 107-121, 1987.
2. John L. Hennessy, David A. Patterson, Computer Architecture A Quantitative Approach, Morgan Kaufmann Publishers, San Mateo, 1990.
3. R.R. Oehler and R.D. Groves, "IBM RISC System/6000 processor architecture", *IBM Journal of Research and Development*, Vol. 34, No. 1, pp. 23-36, 1990.
4. Michael J. Flynn, Chad L. Mitchell and Johannes M. Mulder, "And Now a Case for More Complex Instruction Sets", *IEEE Journal on Computer*, pp. 71-83, 1987.
5. Norman P. Jouppi, David W. Wall, "Available Instruction-Level Parallelism for Superscalar and Superpipelined Machines", *IEEE Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, Mass, pp. 272-282, 1989.
6. Michael Butler, Tse-Yu Yeh, Yale Patt, Mitch Alsup, Hunter Scales, Michael Shebanow, "Single Instruction Stream Parallelism is Greater than Two", *ACM Proceedings of the 18th Annual International Symposium on Computer Architecture*, pp. 276-286, 1991.
7. Michael D. Smith, Mike Johnson and Mark A. Horowitz, "Limits on Multiple Instruction Issue", *IEEE Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, Mass., pp. 290-302, 1989.
8. Kui Yi, Yue-Hua Ding, "32-bit RISC CPU based on MIPS" , *Proceedings of Second Pacific-Asia Conference on web mining and web-based application*, pp. 124 - 128, 2009.
9. Bassoy, C.S.,Manteuffel, H., Mayer-Lindenberg, F.,"SHARF: An FPGA-Based customizable processor architecture", *Proceedings of International Conference on Field Programmable Logic and applications*, pp.516-520, 2009.
10. Gautham,P., Parthasarathy, R., Balasubramanian, K., "Low-Power Pipelined MIPS Processor Design", *Proceedings of the 12th International Symposium on Integrated Circuits, ISIC*, pp. 462 – 465, 2010.
11. Adamec, F.,Fryza, T., "Design and Optimization of ColdFire CPU Arithmetic and Logic Unit", *Proceedings of 16th International Conference on mixed design of integrated circuits & Systems*, pp. 699 – 702, 2009.
12. Shofiqul Islam, Debanjan Chattopadhyay, Manoja Kumar Das, V Neelima; Rahul Sarkar, "Design of High Speed Pipelined Execution Unit of 32-bit RISC Processor" *India Conference, Annual IEEE* , pp. 1 - 5, 2006.
13. Geun-young Jeong; Ju-sung Park; Science and Technology, " Design of 32-bit RISC Processor and efficient verification" 2003, *Proceeding of the 7th Korea-Russia International Symposium*, vol.2, pp. 222 - 227 , 2003.
14. J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan & Kaufman Publishers, San Mateo, California.
15. G.M.Amdahl, G.A. Blaauw, F.P. Brooks, "Architecture of the IBM System/360, *IBM Journal of Research and Development*, vol.8, pp.87-101, April 1964.

AUTHOR PROFILE

Preetam Bhosle obtained Bachelor of Engineering in Electronics and Communication Engineering from Appa Institute of Engineering and Technology, Gulbarga, VTU, Belgaum in 2010 and now pursuing Master of Technology (SP&VLSI) in Electronics and Communication Engineering, from School of Engineering and Technology, Jain University, Bangalore-Karnataka, India.

Hari Krishna Moorthy obtained Bachelor of Engineering from G.V.I.T. K.G.F, Bangalore University in 2001 and Master of Engineering from Sathyabama University, Chennai in the year 2007. Assistant Professor in the Department of Electronics and Communication Engineering, School of Engineering and Technology,Jain University, Bangalore-Karnataka, India.