Low-Power Design Of MIPS Processor

Kedari Theja Katta: UFID- 62404999; Lalitha Mathi UFID- 29341966;Rahul Reddy Chamala: UFID- 41961127; Zhaoyuan Liang: UFID- 19012241

Department of Electrical and Computer Engineering, University of Florida

Abstract— Power consumption is one of the major themes in the present IC design making it as important as performance and area. This paper presents the design and implementation of low power 16-bit Microprocessor without Interlocked Pipeline Stages (MIPS) processor for based on pipeline architecture. We have utilized the Verilog Hardware Description Language (HDL) for the implementation of power efficient MIPS architecture using the Clock gating technique that reduces unwanted clock transitions and stall power reduction. This leads to significant power reduction compared to a baseline MIPS processor. We have implemented and synthesized our Verilog code to evaluate each module and their functionality, power dissipation, latency, area; carried out a placement and routing and generated schematic/layout of the MIPS processor. The layout has been verified for DRC and LVS errors.

Keywords— MIPS, RISC, parallel pipelined, VHDL, clock gating.

I. INTRODUCTION

MIPS architecture is a basic five stage architecture based on RISC(Reduced Instruction Set Computer) implemented on a single VLSI chip. MIPS has a wide range of applications. The implementations of MIPS may differ even though the architecture is the same. High Speed MIPS architectures are pipelined for improvement in performance and frequency. The architecture of MIPS consists of instruction set, registers, adders, layout etc. and the hardware implementation refers to the manner in which different processors use the architecture. We have implemented baseline MIPS architecture and compared it to a parallel pipeline model that we have designed and implemented.

The Five Stages of the MIPS architecture are: Instruction fetch (IF); Instruction Decode (ID); Instruction Execution (EX); Memory Access (MEM) and Write back (WB). We can measure performance based on throughput and Cycle per Instruction (CPI). Pipelining is the overlap of multiple instruction executions. It allows the parallel execution of the different stages of subsequent instructions improving the number of instructions executed in a shorter time. In other words pipelining does not wait until the end of an instruction to begin the next instruction. Although this improves the performance greatly, it also gives scope to hazards that prevent the execution of the next instruction during the designated clock cycle. Hazards are of three types.

a) Structural hazards arise when the flow of instructions requires more hardware resources than those available on the platform [1].

b) Data hazards arise when there is a data dependency between the current instruction and the previous instruction in the pipeline.

c) Control hazards arise when there is a change in the flow of the program such as a branch or a jump instruction that changes the PC.

II. MIPS PROCESSOR ARCHITECTURE

MIPS 16-bit instruction set provides 8 internal registers and has three types of instructions: ALU instructions; Load-Store Instructions; Branches and Jumps. It can also be classified into Register, Immediate and Jump.

The pipeline stages are shown in the following figure.

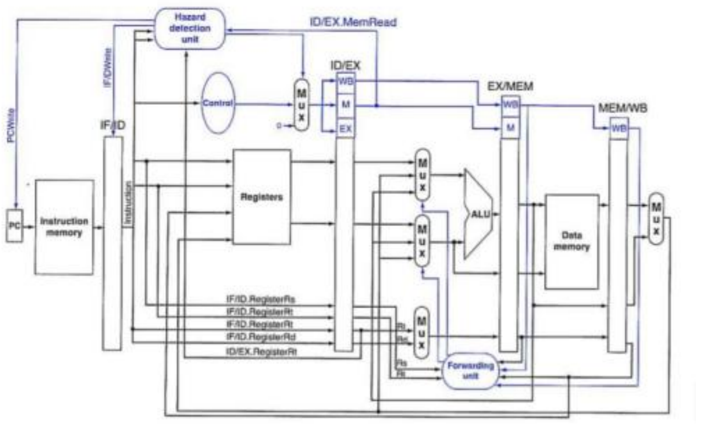

Fig 1 Pipelined MIPS architecture with five stages

FIVE STAGES OF MIPS PIPELINE

a) Instruction Fetch stage (IF): During this stage, the program counter (PC) accesses instruction memory and fetches the next instruction to be executed. The instruction fetched is then passed on to Instruction decode unit.

b) Instruction Decode (ID): In this stage, the instruction is decoded and the register operands required are obtained.

c) Instruction Execution (EX): In this stage, for R type instructions, the ALU instructions are executed based on

ALU control signals while in load and store Instructions, effective address calculation is done.

d) Memory Access (MEM): In case the instruction being executed is load or store, then memory is accessed for obtaining the data. The load and store instructions write to and read from the memory in the memory stage on the basis of effective address calculated.

e) Write-back stage (WB): In this stage, the calculated results from the Instruction execute stage or from the Memory access stage are written back into the registers in the register file. This value is retrieved based on the instruction. For arithmetic instructions the result is taken from the execute stage, whereas for the load and store, result is taken from the memory access stage.

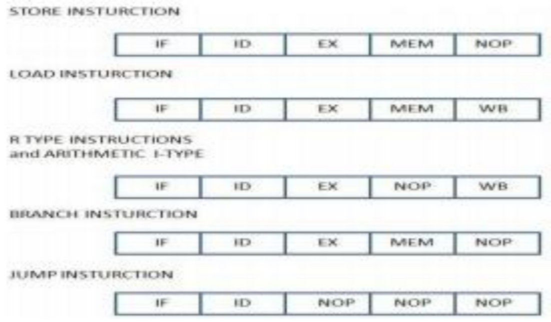

Fig. 2. Pipelined Representation of Instructions

III. DESIGN AND IMPLEMENTATION METHODOLOGY

Our design includes these basic modules
1) Instruction Memory and Data Memory.
2) Datapath
3) Control Logic
4) Hazard Detection Unit
5) Power Reduction Unit.

Instruction memory and Data Memory: The Instructions that are to be processed are stored in Instruction memory and retrieved as required. It is 16 x 256 bits wide and has an input that is an 8 bit address from the program counter while producing an output of 16 bit instruction word. Data memory is for the load and store instructions and it is 8 x 256 bits wide. Additionally it also has a register file in the decode stage made of thirty two 8 bit registers.

Datapath: The datapath consists of the 5 stage pipelined structure which are Instruction fetch (IF); Instruction Decode (ID); Instruction Execution (EX); Memory Access (MEM) and Write back (WB). Also registers are placed between each stage of the pipeline to forward the result of the previous stages to the next.

Control unit: The control module interfaces all the components of the processor by means of generated signals that are used for coordinating required components of entire processor. Thus both flow control of data amidst the pipeline and signal generation for hazard detection are mainly carried out by the control unit.

Hazard Detection unit: Hazard detection unit observes the hazards and determines when forwarding cannot occur. In order to execute the dependencies serially, it stalls the pipeline at the required stages for a few clock cycles.

Power reduction techniques in our implementation:
Low power Design has been of importance because of factors, such as the emergence of portable systems where in it helps to have less power dissipation, Reducing power dissipation reduces the heat dissipation as well increasing the battery life and reliability of the chip. With the increase in demand for power efficient and environmental concerns, low power design is a rising area. We know that using a pipelined processor improves speed and throughput. Thereby this design is used in many applications where speed and throughput are of concern. The main power reduction techniques we have implemented in our MIPS design are

Stall Power Reduction:
Different types of instructions require different stages and execute differently. For example ALU instructions do not use Memory access (MEM) stage while Store instructions do not need a Write back (WB) stage. Load instructions though go though all stages across a pipeline. When a stage is not used by an instruction it causes unnecessary power consumption. To avoid this, a by-pass line is used to skip the unused stages across the pipeline. This is called Stall power reduction. In arithmetic instruction MEM stage is not used, so the value obtained from the Instruction execution (EX) stage is directly forwarded Write back(WB)stage. Also the EX/MEM pipeline registers are at zero so that no transitions occur. For a Load-Store instruction clock is disabled for the MEM stage unless it is a store instruction and thus the power dissipation is reduced.

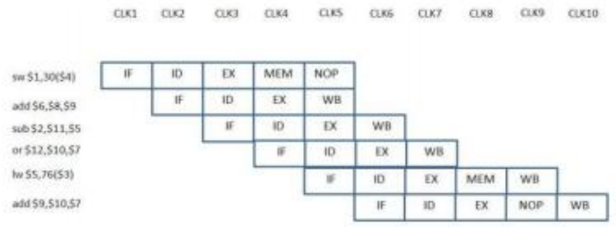

Fig. 3. Reconfigured Pipeline with Bypassing

This bypassing of data can be continued until there is a Load instruction which uses all five stages of the pipeline causing a resource conflict shown by crossed-arrows in Fig. 5. So, data has to pass through the regular pipelined structure till it encounters a different instruction for which the modified bypass pipeline can be used.

Clock Gating:

Clock gating is a power optimization technique used in this project wherein power reduction is achieved by disabling the pipeline stages when a stall is inserted into the pipeline. It achieves the power reduction by disabling the pipeline stages that cause unnecessary switching activity. One of the influencing factors of dynamic power dissipation is switching activity where the dynamic power is given by the equation,

$$P = 0.5*C*(Vdd)2 *\alpha sw*fclk$$

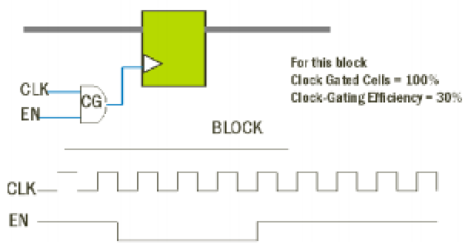

Fig 4. Simple Clock gated register

Thus by reducing switching activity, αsw, unwanted transitions are reduced which reduce the dynamic power consumption.

IV. IMPLEMENTATION

We have implemented an 16 bit pipelined MIPS processor stages in Verilog hardware description language. Each stage of the pipeline has been implemented as an individual module. All the modules are placed under a top module "mips_16_core_top" which interconnects the various stages (IF, ID, EX, MEM, WB). Hazard detection is carried out and if there is no hazard then instruction fetch is enabled. The rest of the stages follow. The functionality is tested on Xilinx after which Simulation; synthesis; Timing and Power analysis; Floor Planning, Placement and Routing are carried out.

The list of instructions implemented in our processor are shown in table 1

Instruction type	Instruction
R-type	ADD, SUB, ADDI, SLT, SRL
I-type	LW, SW
J-type	BEQ, JUMP

Table1. Instructions that are implemented in MIPS

Stall power reduction is carried out as mentioned earlier. The clock gating is implemented in two stages.First, the clock is gated with reset signal in the top module "mips_16_core_top" that when reset is high, the clk_p (gated clock) keeps 0 to avoid unnecessary transitions. Secondly, the clk_p from the top module is gated with enable signals in each stage module that when a stall from hazard_detection module comes in, the IF&ID stage will be frozen and the register of pipeline data will keep 0 that no transition will be done.

V. CONCLUSION AND RESULTS

In this project we have successfully implemented a 16 bit MIPS with a hazard detection unit. We have optimized it with the techniques clock gating, stall power reduction techniques and compared it to a baseline code. Simulation on Xilinx for a pipelined processor for ALU operation multiply has been tested and given as

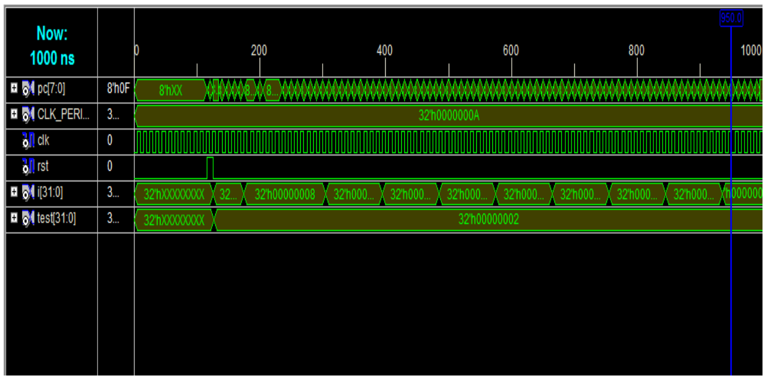

Fig 5. Simulation in Xilinx for multiply in MIPS16

We obtained timing and power analysis from the timing.rep and power.rep files generated during synthesis. The reduction in power compared when clock is gated with stall in all the modules compared to baseline code with no optimizations is found to be 49.39%. The reduction in power with two stages of clock gating as compared to the baseline code is found to be 62.52%.

By comparing the optimized and unoptimized power, we get the following results.

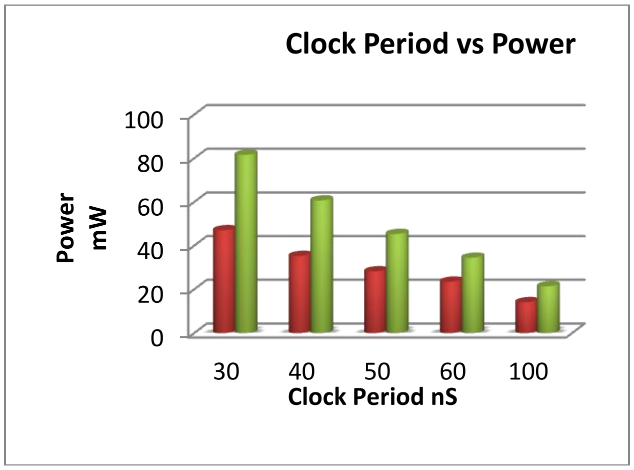

Fig 6. Power for unoptimized and optimized MIPS processor at different time periods

Comparing the pipelined code for different frequency we get the following graph

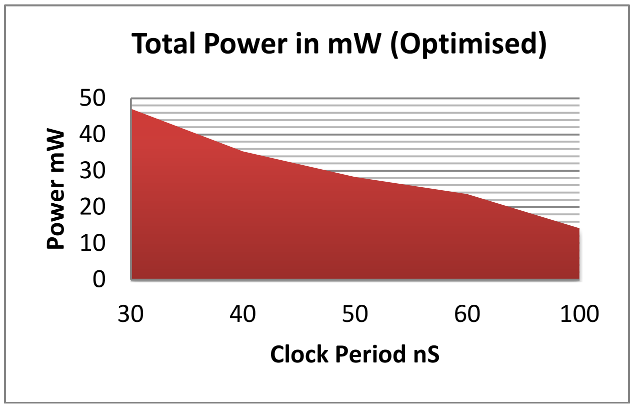

Fig7. Power dissipation in the optimized code for different frequencies

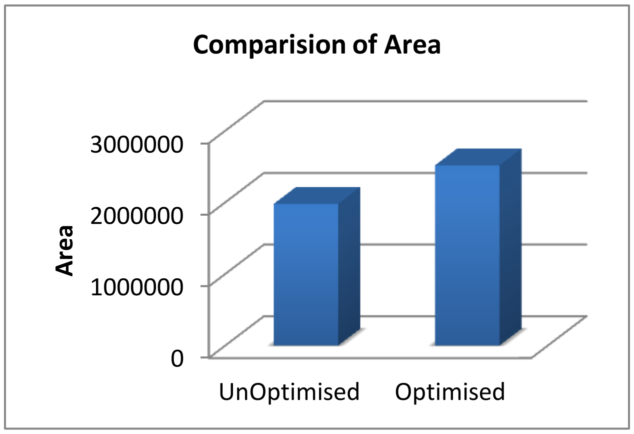

Fig 8. Area comparison

The DRC and LVS checks are completed and the layout. The area comparision is given above. The layout of the optimized design is shown in the following figure.

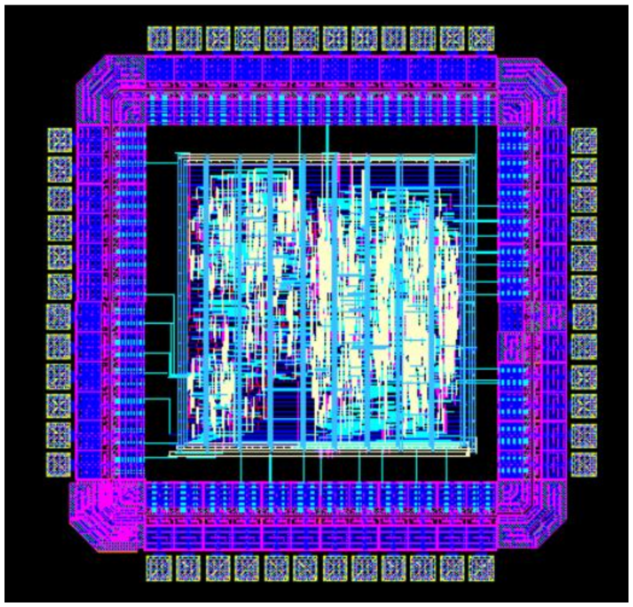

Fig 9. Final layout of optimized MIPS processor with IO pads

VI. REFERENCES

[1] Computer Architecture, Fifth Edition: A Quantitative Approach (The Morgan Kaufmann Series in ComputerArchitecture and Design)

[2] Low power pipelined MIPS processor design- Gautham P, Parthasarathy R, Karthi Balasubramanian Department of Electronics and Communication Amrita Vishwavidyapeetham

[3] “Low-power CMOS digital design,” A. Chandrakasan, S. Sheng, and R. Brodersen, IEEE J. Solid-State Circuits, vol. 27, pp. 473–483, Apr. 1992.

[4] A FPGA Implementation of a MIPS RISC Processor for Computer Architecture Education ,By Victor P. Rubio, B.S. Master of Science New Mexico State University Las Cruces, New Mexico [5] VHDL

[5]Implementation of a MIPS RISC processor Anjana R and Krunal Gandhi, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2,Issue 8,August 2012