Pipelined MIPS With Improved Datapath

Harpreet Kaur, Nitika Gulati
Research scholar, Electronics and Communication Engineering, RIEIT, Railmajra (Punjab), India
Assistant professor, Electronics and Communication Engineering, RIEIT, Railmajra (Punjab), India

ABSTRACT
This paper proposes a five stage pipelined processor with reduced number of unwanted transitions due to the stalls present in the pipeline which results in the reduction of dynamic power. To reduce the unwanted transitions, modification in the datapath is proposed. The proposed architecture includes the instruction and data memory, register file, datapath, control unit, data forwarding and hazard detection unit. The processor architecture is described using verilog and synthesized using Xilinx Spartan 3E.

Keywords: MIPS processor, pipeline, writeback, stall.

I.INTRODUCTION

With increasing demand for low power battery driven electronic systems, power efficient design is presently an active research area. Batteries contribute a significant fraction of the total volume and weight of the portable system. Low power processors are the key to the realization of the portable electronic devices, in which power consumption is an important factor. MIPS is a general purpose five stage pipelined micro architecture based on RISC design principle designed to be implemented on a single VLSI chip.
MIPS is a load/store architecture i.e. data may be operated on only when it is in a register and only load/store instructions access memory. With pipelined structure some hazards get introduced that results in incorrect computation. There are typically three types of hazards which are data, structural and control hazards. Data hazard may occur when an instruction scheduled blindly, would attempt to use data before the data is available in the register file. Structural hazard occurs when a part of the processor's hardware is needed by two or more instructions at the same time. The last hazard which is control hazard occurs when a branch prediction is mistaken. To eliminate these hazards processor frequently includes NOP instruction to the pipe. But NOP do not do useful work. Therefore power may be dissipated if data transitions take place and the percentage of dynamic power consumed by NOP instruction in a pipelined processor is considerable. Dynamic power depends upon the switching activity or in general number of transitions and is given by equation:

$$P=\frac{1}{2}C_0(V_{in})^2Nf$$

Thereby decreasing number of transitions (N) results in reduced dynamic power consumption.
The aim of this paper is to reduce the dynamic power consumption of pipelined processor by reducing unwanted transitions.

II.INSTRUCTION SET OVERVIEW

The purpose of an instruction is to specify both an operation to be carried out by a processor and the set of operands or data to be used in the operation. There are three types of instructions supported by

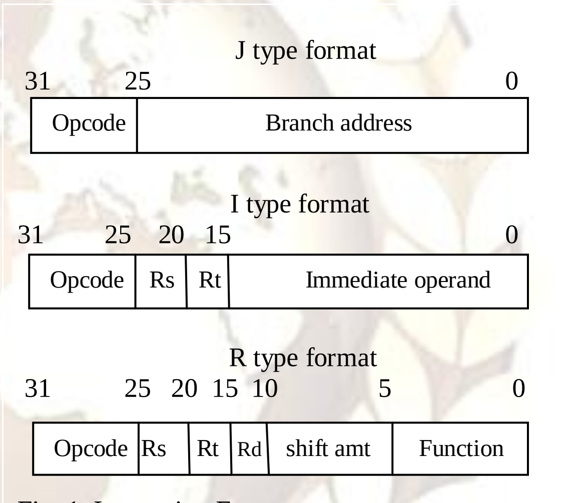

Fig. 1. Instruction Format

MIPS processor. These are register type, immediate type and jump type. The format for these instructions is shown in fig. 1. All the instructions are 32 bits in length and contain 6 bit opcode in a fixed position. The remaining 26 bits are used in various ways, depending on the instruction type.
In case of jump type instruction, the 26 operand bits form a memory address, which is target or branch address. The immediate type and register type formats specify register addresses using two and three 5-bit fields respectively. Since the register addresses occupy only 15 bits of the instruction format, the remaining 11 bits are used in various ways to increase the range of operations that can be performed.

III.MOTIVATION

The need for low power design is motivated by several factors, such as the emergence of portable systems, thermal considerations, reliability issues and environmental concerns. Low power consumption helps to reduce heat dissipation, lengthen battery life, and increase device reliability. In battery powered applications, where speed is less of a concern, pipelined processors are often used. The pipeline stages of MIPS for different type of instructions are shown in fig. 2.

STORE INSTRUCTION

IF	ID	EX	MEM	NOP

LOAD INSTRUCTION

IF	ID	EX	MEM	WB

sR TYPE and AIRTHMETIC I TYPE

IF	ID	EX	NOP	WB

BRANCH INSTRUCTION

IF	ID	EX	MEM	NOP

JUMP INSTRUCTION

IF	ID	NOP	NOP	NOP

Fig. 2. Pipelined Representation of Instructions

It can be seen that arithmetic type instruction do not use memory access stage. Store instruction do not require write-back stage while load instruction go through all pipeline stages. Transitions during the unused stage cause extra power consumption. These unwanted transitions can be reduced by by-passing the unused pipeline stage. In arithmetic instruction memory access stage is not used, so data obtained from the execution stage is forwarded directly to write back stage.

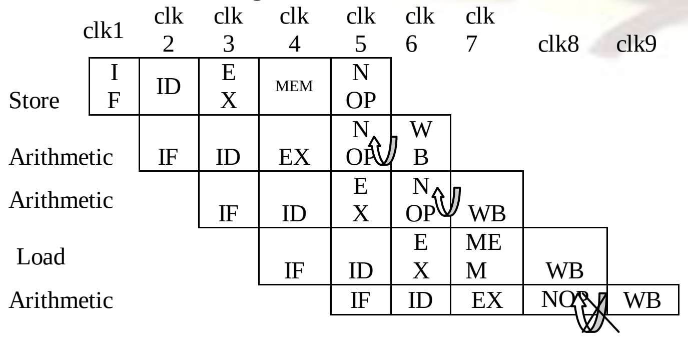

Fig. 3. Pipelining with Bypassing

In the earlier research, the bypassing of memory access stage for arithmetic instructions occurs after a store/branch/jump instruction. The store instruction has NOP stage during write back stage while arithmetic instruction have NOP stage during memory access stage. Hence write back stage of arithmetic instruction can be moved to the memory access stage without any resource conflict as shown in fig. 3. This bypassing of data can be continued till the load instruction is encountered.

In the scenario if arithmetic instruction appears after load instruction the only possible way to overcome the conflict was to insert a NOP to the arithmetic instruction before its writeback stage. The idea presented in this paper is to overcome this problem of inserting NOP instruction, as it leads to delay of one cycle and the transition of signal leads to power consumption. This is done by bypassing a memory access stage even after load instruction is encountered as shown in fig. 4

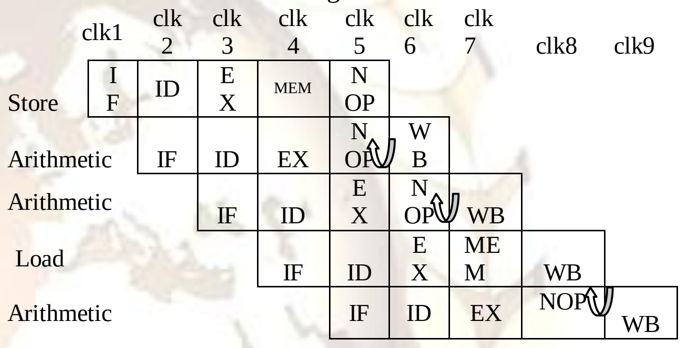

Fig. 4. Reconfigured Pipelining with Bypassing

IV. PROPOSED ARCHITECTURE

The proposed pipelined architecture of the processor is shown in fig. 5. It includes the instruction and data memory, register file, datapath, control unit, data forwarding unit and hazard detection unit

Instruction memory: The instruction memory contains the instructions that are executed by the processor. It is 32 by 1024 bytes wide and takes 32 bit address from the program counter as an input and gives 32 bit instruction word as an output.

Data memory: Data memory is accessed by the load and store instructions and it is 32 by 256 bytes wide.

Register file: Register file contains thirty two 32 bit general purpose registers. Generally it has two read ports and one write port but in this proposed architecture register file has two read ports and two write ports and therefore can perform two simultaneous read and write operations.

Datapath: The datapath consist of 5 stage pipelined structure. The five stages are fetch, decode, execute, memory access and write back stage. Pipeline registers are placed between each stage and they are used to carry the result of the previous stage to the following stages.

Fetch stage: In this stage, the content of the program counter is used to access memory and fetch the next instruction to be executed.

Decode stage: During this stage, the instruction is decoded and the required operands are retrieved and the opcode is passed to the control unit which asserts the required control signals. The idea of implementation of dual write port is achieved by a signal generated during the instruction decode stage.

Execute stage: All the instructions fetched through the instruction memory are computed in this cycle and the data generated during this cycle is passed on to the next pipeline. This cycle includes

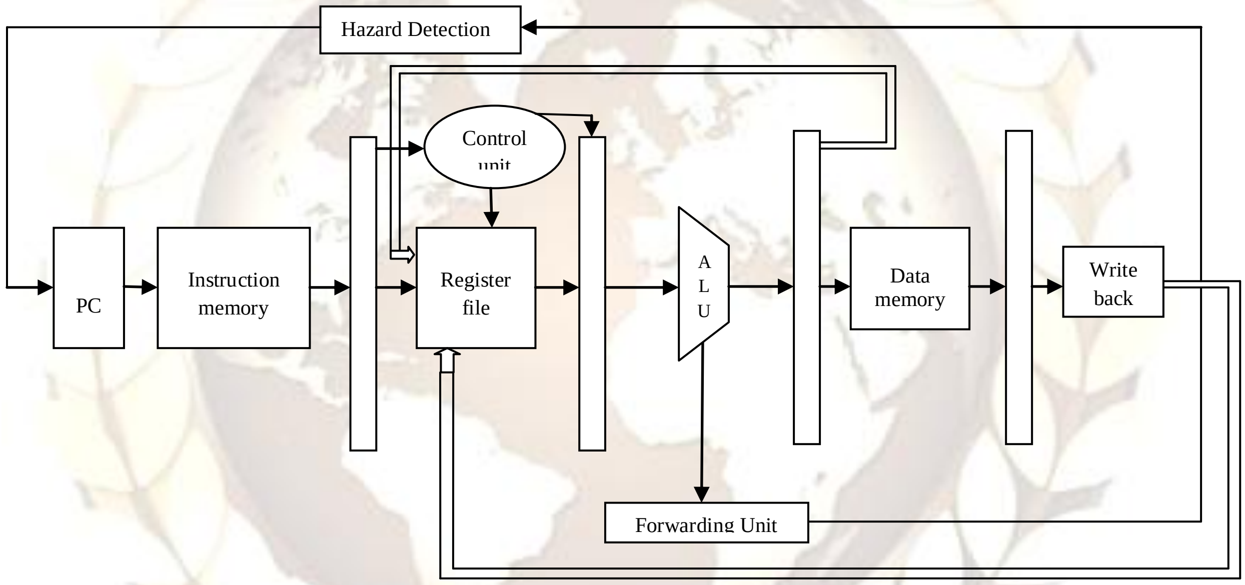

Fig. 5. Proposed MIPS architecture

effective address calculation for load/store instructions. Whenever load instruction is encountered a dual signal is asserted high in decode stage which is responsible to facilitate dual write back for register type instruction following load instruction and the write back value is taken directly from alu result and are written back to the register file using second data bus. This second data bus is powered up only when the dual signal is high. So execute stage decides whether the data should be written back to register file directly from ALU or through writeback stage.

Memory access stage: If the instruction being executed is of the load or store type, then the data memory is accessed during this stage. The previously calculated effective address is applied to the data memory and the read or write operation is performed in accordance with instruction type.

Write-back stage: During this stage, the results of the calculation from the execute stage or from the memory access stage are updated into the registers in the register file. The updated value is fetched in accordance with the type of instruction. For arithmetic instructions the value is taken directly from the execute stage, whereas for the load instruction this value is taken from the memory access stage.

Control unit: The control unit is responsible for generating signals that are used for coordinating all components of entire processor. Moreover the dual signal which is necessary for dual writeback operation is also generated by the control unit.
Data forwarding unit: This unit is responsible for the proper flow of data. Forwarding is implemented by feeding back the output of instruction into the previous stage of the pipeline as soon as the output of that instruction is available.

Hazard Detection unit: This unit detects conditions under which data forwarding is not possible and stalls the pipeline for one or two clock cycles so that instructions can be executed in the correct sequence.

V.RESULT

After modeling the design of 32 bit MIPS through verilog, it was synthesized using Xilinx Spartan 3E. Power analysis was done using Xpower estimator. The design had a maximum frequency of operation of 193.98Mhz. Fig. 6. shows the results of power analysis.

Clock (MHZ)	Toggle rate	Power consumption in normal pipeline (in W)	Power consumption with single write back (in W)	Power consumption with dual writeback(proposed work) (in W)
194	100%	1.359	1.139	1.064
150	75%	1.022	0.857	0.659
100	50%	0.686	0.577	0.340

Fig. 6. Table for Power Consumption Comparison

VI. CONCLUSION

In this work, we proposed a method for minimizing unnecessary transitions by reducing stalls in the pipeline. The proposed approach utilized dual write port register file. The processor was successfully designed in verilog HDL, simulated with Modelsim and synthesized on to a Xilinx Spartan 3E.

REFERENCES

1 D.A.Patterson and J.L.Hennessy, Computer Organization and Design, The hardware/software interface. Morgan Kaufmann, 2005.

2 Gautham P, Parthasarathy R, Karthi Balasubramanian, "Low Power Pipelined MIPS Processor Design," in the proceedings of the 2009 12th international symposium,2009 pp. 462-465.

3 A. A. S. Pejman Lotfi Kamran, Amir Mohammad Rahmani and A. A. Kusha, "Stall power reduction in pipelined architecture processors," in Proceedings of the 21st International Conference on VLSI Design, 2008, pp. 541-546.

4 Xia Li, Longwei Ji, Bo Shen, Wenhong Li and Qianling Zhang, "VLSI Implementation of a High Performance 32 bit RISC Microprocessor," International Conference on Communications, Circuits and Systems and West Sino Expositions, IEEE 2002, Vol. 2, pp. 1458-1461.

5 Zhenyu Gu, Zhiyi Yu, Bo Shen and Qianling Zhang, "Functional Verification Methodolgy of a 32 bit RISC Processor," International Conference on Communications, Circuits and Systems and West Sino Expositions, IEEE 2002, Vol. 2, pp. 1454-1457.

6 M. S. I. Mamun Bin Ibne Reaz and M. S. Sulaiman, "A single cycle mips risc processor design using vhdl," in Proceedings of IEEE International Conference on Semiconductor Electronics, Dec 2002, pp. 199-203.

7 V. Venkatachalam and M.Franz. "Power reduction techniques for microprocessor systems," ACM Computing Surveys, September 2005, vol. 37, no. 3, pp. 195-237.

8 A. Correale, " Overview of the power minimization techniques employed in the IBM PowerPC 4xx embedded controllers," in proc. Of the ACM/IEEE International Symposium on Low Power Design, April 1995, pp. 75-80.

9 R. Razdan and M.D. Smith, "A High Performance Micro architecture with Hardware Programmable Functional Units," Proc. Micro-27, IEEE Computer Society ,1994, pp. 172-180.