**ECE 585-02**

**Research Project**

**“Faster CPUs with New Methods solving Control Hazards in Pipeline Datapath Architecture”**

**Due Date: 12/05/2021**

**Professor: Won-Jae Yi**

**Nader Alnatsheh**

**A20452265**

**A. Acknowledgment: I acknowledge all works including figures, codes and writings belong to me and/or persons who are referenced. I understand if any similarity in the code, comments, customized program behavior, report writings and/or figures are found, both the helper (original work) and the requestor (duplicated/modified work) will be called for academic disciplinary action.**

**Signature:** Nader Alnatsheh, A20452265, Date: 12/051/2021

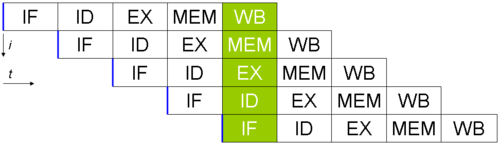
**Abstract:** the purpose of this research project is to get a deeper understanding on the importance of the pipeline datapath architecture, and how it is necessary to find solutions to the disadvantages that it has, like the hazards, so it always stays our favorite datapath implementation. And so, we engineers get to design Faster CPUs. Three new methods have been presented and described to solve the control hazard problem that occur in the pipeline datapath architecture. With the main focus of the papers is to prevent the control hazard to a percentage of 0%, with no matter how long the assembly code is or the high number of conditional branch instructions there are. The three papers have been published after 2010, which is pretty new techniques in the field of eliminating control hazards. The three papers presented new methods that solved the control hazard using Hardware modifications, and that’s mean extra hardware/components to be added which will be the one disadvantage for all of them, in terms of the Area, Power consumption and the propagation delay, and here comes the beauty of VLSI in implementing such techniques that will help to design these new methods.

**Introduction:**

**Pipelining** is an implementation technique that is used to execute multiple instructions in parallel with a delay of one clock cycle between each one of them. Most of today Computers CPUs are based on pipelined processors. The main advantage of pipelining technique is that the performance of the processor is increased “faster processors”, and the reason of that because every hardware part of the processor is busy, which increase the amount of the useful work the processor can do in a specific time. A clear line has to be addressed here is that Pipelining do not reduce the time it takes to complete a one individual instruction, but instead it increase the number of instructions that can be executed and completed at the same specific time, which means that pipelining improves the overall instructions throughput instead of the one instruction execution time.

Figure [1] shows an overview of Instructions in pipelined processors that get executed in steps called stages. In MIPS pipelined architecture there are five different stages:

1. **Instruction Fetch (IF):** where the instruction gets fetched from the memory.
2. **Instruction Decode (ID):** deciding what type of instruction and what does it mean.
3. **Execution (EX):** execute the instruction or calculating the address. “ALU job”.
4. **Memory Access (MEM):** Store or load operands into or from the memory.
5. **Write Back (WB):** Write the result back into the register.



**Figure [1].**

The main disadvantages of implementing pipelining technique when designing processors is that it is complex to design them, the instruction latency is increased, the throughput is hard to predict, and the most important is the potential of hazards to occur.

Hazards are simply known as the inability of the next instruction to get executed in the next following clock cycle. There are three different types of hazards in pipelined processors:

1. **Control Hazards:** the pipeline processors keep doing the job with no stalls as long as the address of the next instruction is known, but once a branch instruction gets fetched, the next instruction address is not known until the branch instruction get executed first.Means that the processor will have to wait until the correct location of the next instruction is known before fetching more instructions. This situation is known as control hazard or also known by the name branch hazard. **This research paper represents three new solutions to solve the control hazards in pipelined processors.**
2. **Data Hazards:** is simply known as the situation when an instruction can’t be executed in the proper clock cycle, this happen when the result of one instruction operation is going to be used in the following instruction as an operand, but its value is not calculated yet!
3. **Structural Hazards:** is a situation that occur when two instructions need the same

hardware at the same specific clock cycle. One way to solve this is by using the Harvard Architecture instead of the von Neumann Architecture, where the instruction and the data are stored in a two different storage and have two separate busses (signal path).

**Problem Statement:** as I indicated earlier with pipelining, the number of instructions that get executed simultaneously is increased which means increasing the overall performance of the computer CPU which is the biggest advantage we can get from the pipelining techniques. But on the other hand, there are some disadvantages of using pipelining, Hazards are one of them, it is fair to say that the longer the pipeline, the worser the problem of control/branch hazard instruction will get. There are a lot of methods that solved the control hazards, the most famous one is Prediction, the idea of it is to always predict that the branch will be untaken, if we were right on that guess, the pipeline technique proceeds at full speed, but if not and the branch was taken, a stall will occur. Statistical, fuzzy, and leaning method based on neural networks is all methods that could solve the control hazard problem, but sometimes they can not prevent it. Next, I will introduce three methods that solve the control hazard 100%, all of them focuses on finding a solution using the hardware direction.

**Three papers that presented new methods to prevent Control Hazards:**

**The first paper is “**A new method to prevent control hazard in pipeline processor by using an auxiliary processing unit” [1].

**The proposed solution:** an auxiliary processing unit was added to the computer datapath architecture and implemented to prevent control hazards and also, it was modified to determine if its path of instructions will have a conditional branch instruction in it or not in the future. Figure[1] shows an overview of the new datapath architecture with the new auxiliary processor unit added, here let us start by initially saying we will have 2 processors A and B, with the auxiliary processor unit as B and the main processor as A; the main processor will fetch all the instruction from the register file but not the branch conditional instruction and it will keep detecting if there will be an upcoming next branch instruction, here the auxiliary processor unit “B” will be disabled. Next, when the main processor “A” detects a conditional branch instruction, it will keep fetch instructions from the path that does not have a conditional branch instruction in it, the job of the auxiliary processor unit “B” here is to fetch the instructions from the path that have a conditional branch instruction. Now when the branch instruction gets executed and by the end of the EX-stage, it will be known which of the 2 processors (A or B) followed the path of the branch instruction. Now after knowing which processor unit choose the path with no conditional branch instruction becomes the **main processor “A”** and so go on and fetch and execute the next instruction in order and the other unit become disabled and stay as an **auxiliary unit ‘B’.**

Diagram

Description automatically generated

**Figure [2].**

**Results:** the paper didn’t show any implementation of the proposed solution, and so there was no results, but the proposal solution theoretically was an excellent idea.

**The second paper: “**Resolving Control Hazard by Reconstituting Processor's Pipeline Organization”[2].

**The proposed solution:** the paper reconstructed the regular known pipeline architecture datapath with two proposals that focuses on modifying the Program Counter (PC). **The first modification** is that the first clock cycle will be completely used only by the IF stage to fetch new instruction, this is done by set the PC initially to a value of ‘mins 1’ instead of ‘0’ and so by the end of the first clock cycle, the address of the first instruction will be pointed to index zero in the Instruction memory and will be ready to get fetched, and at the beginning of the second clock cycle, the fetched instruction will be transferred into the ID stage and so on to get executed. **The second modification** was changing the use of the PC, as in the regular pipeline architecture the PC holds the address of the next instruction that is going to get executed in the next clock cycle, but here the PC holds the address of the same instruction that is being decoded in the ID stage, and after that the address will be incremented by one to get the address of the next instruction. (PC + 1). Figure [3] below shows the execution pathway of the proposed design.

Diagram, schematic

Description automatically generated

Figure [3].

In the Instruction decode stage (ID), specifically at the beginning of the second clock cycle, the ID stage will contain an adder, which the inputs of it will be an Immediate value from the IF/ID Inter- stage register and the value of the PC register. Also, it will contain a branch detection circuit. The main change in this stage is that there will be no Next PC input (NPC) from the IF-ID inter stage register going to the ID stage.

**Results:** two tests have been implemented to try the new proposed architecture, with the first code having 16 lines, with three branch instructions, the first two were taken and the last one was not taken, the results shown no stalls. The second test was a total of 65280 branch instruction were all handled with zero misprediction per 100 instruction. A comparison has been done between the proposed architecture and three processors of Intel which are i3,i5 and i7. The result is shown in the table [1] below with the proposed architecture have 0.0% branch misprediction which means that the processor will never have Control Hazards.

Table

Description automatically generated

Table [1]

**The third paper is “**An Improved Pipelined Processor Architecture Eliminating Branch and Jump Penalty”[3].

**The proposed solution:** the paper presents a new architecture that eliminates the branch and jump penalty and reduce the cost of calculating the destination address. Two new proposal was described, with the first was the modification of the I-format instruction “*Branch Instruction Format*”. In MIPS architecture, if the branch was Taken, the calculation of the target address “New PC value” will be *PCnew = PCold + offset (SE & sll>>2)*. A new architecture of the I-format was developed to reduce the cost of the destination address which will be determined by the segment number and segment offset. Table [2] below shows the new architecture.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Op | Rs | Rt | Selection | Segment offset |
| 6 bit | 5 bit | 5 bit | 2 bit | 14 bit |

Table [2].

The least significant two bit of each instruction address will always be 00, because if we start our PC value initially at 0 then the next value for the PC will be 4 which in binary >> 0100 and the one after is 8 >> 1000 and so on, because of this feature the paper described a new way in presenting the current value of PC, it started with dividing the 16-bits “Immediate-field” into 2 fields the first with a 2-bits calling it by *PCseg*, where, the upper and lower segments of *it will be*  PCseg+1 and PCseg-1 respectively, and similarly the upper value of *PC*+1 will be *PC*±2. Now, the calculation of the branch destination address will be the summer of segment offset from the instruction and the segment number PCseg; which will be one of these values: PCseg, PCseg-1, PCseg+1 or PCseg±2 depending on the selection bit of the branch instruction. Table [3] shows the selection bit.

So, the branch destination address will be = either one of these values: PCseg-1| PCseg | PCseg+1 | PCseg±2], which is a 14-bit offset >>2 by two bit. The values of PCseg-1, PCseg, PCseg+1 and PCseg±2 will be stored in four registers, because when the current segment of PC is increment for a branch or jump destination address then, PCseg-1, PCseg, PCseg+1 and PCseg±2 is updated too. While the extra update will happen here to the PCseg±2 if PC change position from upper half portion of PCseg to lower half portion. Figure [4]: shows an overview of the proposed architecture.

A big advantage here will be that the time required to calculate the destination address is the time required to select a segment from four segments according to the selection bit i.e the propagation delay of a 4 to 1 mux which is very smaller than that of a 32-bit adder used in MIPS. Here our candidate segments are PCseg-1, PCseg, PCseg+1 and PCseg±2 as we have two selection bit.

Table

Description automatically generated

Table [3]

Diagram, schematic

Description automatically generated

Figure [4].

**Results:** the proposed architecture advantage is that it requires no branch predictor and branch target buffer, also it require no time to generate the jump address, because the if jump address instruction got fetched in a clock cycle, then the next clock cycle we will get the address through the *address and jump destination*.

**Discussions/ Opinion:**

The first paper solution of using a second processor which they called it an “auxiliary processor unit” is an excellent method in terms of a theory in eliminating control hazards in the pipeline architecture, what would I loved to see is implementing the design and seeing results for the method, which will support their proposal. Now in terms of VLSI concepts, I see that the area will be increased and that’s mean increase in the cost and the power consumption, I am not sure about the delay here, but I think we will also see a raise in the propagation delay a little bit.

The second paper solution in my opinion was a complete method starting from breaking the regular normal pipeline datapath architecture starting from reconstructing the first and second clock cycles definitions. In terms of VLSI concepts, we will see a raise in power consumption and area, which is not that bad comparing to the first method.

The third paper solution proposal focused more on reconstruct the I- format instruction and specifically dividing the 16-bits Immediate field into two fields, the idea and implementation is different, which I believe in implementing it will be a more complex than the first two methods. In terms of VLSI concepts, the area and the power consumption will be increased, and the propagation delay will be only calculated from the time we have to selecta Pc segments from one of the 4 choices, which is not bad comparing to the first solution proposal.

Table below shows a comparison between the three of them with respect to the normal pipeline architecture (X).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Area | Power consumption | Propagation delay | Eliminating Control Hazards |
| Method 1 | 2x | 1.8x | almost the double | 0% |
| Method 2 | 1.5x | 1.5x | Little to no delay | 0% |
| Method 3 | 1.2x | 1.2x | Faster | 0% |

**Conclusions:** In conclusion, a brief introduction to the pipelining datapath architecture was described, also three new methods were presented to prevent the control hazard to occur with a percentage of 0%. The three papers focused on hardware modifications. The results were presented too, and a discussion and my opinion on the papers was described.

**Future work and thoughts:** after doing this research, I got encountered to a quite few good papers that presents methods to solve the hazards problem in general, I really enjoyed the papers that proposed new implementations of the full adders, with less devices and using the FinFET architectures, because my main focus is VLSI with a lot of them focused on the structural hazards, my topic first was on methods to solve the structural hazard but I figured that would be too much in detail on VLSI, which is not the main focus of the class, so I started looking into methods to solve control hazard.

For future work will be first in the winter break to work on implementing the first paper solution, which is the use of an auxiliary processor unit, and try to master VHDL coding.

**References:**

[1] : A. Habibizad Navin, E. Lahouti, M. Lotfi Anhar and M. k. Mirnia, "A new method to prevent control hazard in pipeline processor by using an auxiliary processing unit," 2010 2nd International Conference on Advanced Computer Control, 2010, pp. 596-599, doi: 10.1109/ICACC.2010.5487130.

[2] : A. Pandey, "Resolving Control Hazard by Reconstituting Processor's Pipeline Organization," 2019 Third International Conference on Inventive Systems and Control (ICISC), 2019, pp. 201-206, doi: 10.1109/ICISC44355.2019.9036414.

[3] : M. R. Hasan, M. S. Rahman, M. Hasan, M. M. Hasan and M. A. Ali, "An Improved Pipelined Processor Architecture Eliminating Branch and Jump Penalty," 2010 Second International Conference on Computer Engineering and Applications, 2010, pp. 621-625, doi: 10.1109/ICCEA.2010.126.