Team Pass It On

Phase Three: Control Unit, Additional Hardware, Data Memory, and AUX MUXs

Tennessee Technological University

ECE 4120 – Fundamentals of Computer Design

Department of Electrical and Computer Engineering

Written By: Josh Adair, Cameron Dempsey, Brantlee Garland, Hunter Goodson, and Randy Wilhoite

Table of Contents

[Modified Block Diagrams: 3](#_Toc512526764)

[Phase Three Objectives: 4](#_Toc512526765)

[VHDL Implementation Elaboration: 4](#_Toc512526766)

[Synthesis Results: 5](#_Toc512526767)

[Test Bench Elaboration: 6](#_Toc512526768)

[Waveform Elaboration: 8](#_Toc512526769)

[TimeQuest Timing Analyzer: 8](#_Toc512526770)

[Phase Three Improvements: 8](#_Toc512526771)

[Pipeline Hardware Overhead: 8](#_Toc512526772)

[Forward and Hazard Units Performance Penalty: 8](#_Toc512526773)

[Forward and Hazard Units Hardware Overhead: 8](#_Toc512526774)

[Individual Contribution: 8](#_Toc512526775)

[Phase Three Conclusion: 9](#_Toc512526776)

## Modified Block Diagrams:

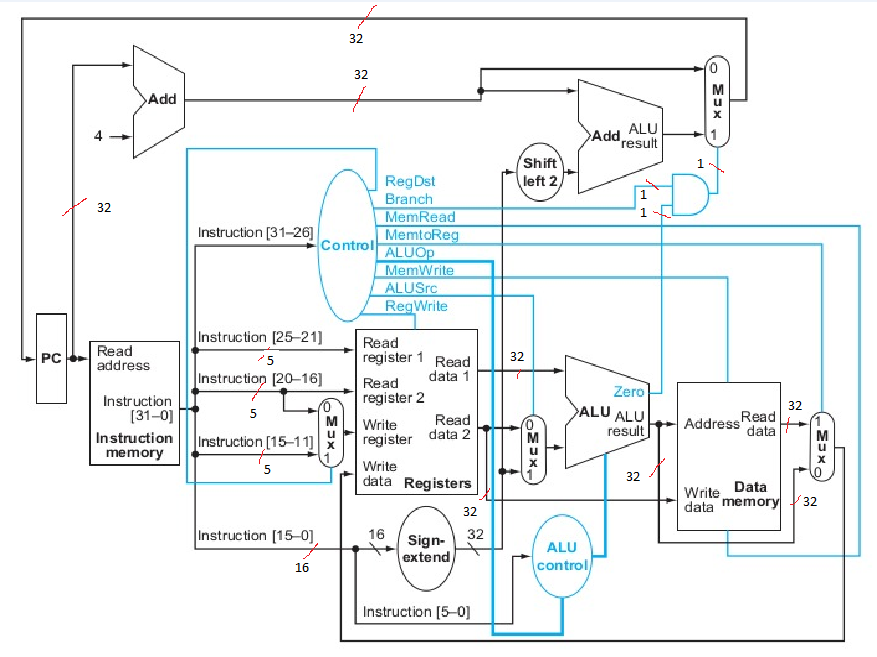


Figure 1. Single Cycle Processor Block Diagram

Figure 2. Pipelined Processor Block Diagram

Figure 3. Pipelined Processor with Forward and Hazard Units

## Phase Three Objectives:

The third and final phase of the project required quite a few additions. A control unit needed to be designed, along with additional branch handling hardware via modifying the ALU. A memory unit for data memory and auxiliary multiplexers were both added in this phase as well. After these units were completed, we needed to combine all of the phases together in order to create single cycle implementation. Next, we would need to implement pipelining into our design. This was achieved using multiple registers. With the required project implementations out of the way, we decided we would create the forward and hazard units for extra credit as well. This involved implementing and demonstrating both stalls and forwarding for data hazards.

## VHDL Implementation Elaboration:

Single Cycle Implementation:

We had to implement multiple stages of the design for this phase of the project. The first stage was for a single cycle processor. In order to achieve this, we have a few Add entities. The first is an entity called Add4, which has two inputs and one output. The first input is A, a 32 bit value from the PC. The second input is a constant. The single output is Q, a 32 bit value. The architecture basically includes a process that does the addition of the inputs to achieve the added Q output. An adder was also needed after the shift left operation. This was performed via an entity called AddShiftLeft, which had two 32 bit inputs and one 32 bit output. The architecture for this adder was similar to the first adder we used; however, it was for the PC offset. One of the adders also needed a multiplexer. This was created with an entity called AddMux. It contained two 32 bit inputs and a 1 bit select. The output for AddMux is a single 32 bit value called X. The architecture involves selecting X = A when select is equal to 0, otherwise X = B. An AND gate was needed for branching. It’s rather simple with two 1 bit inputs called A and B along with a 1 bit output called C.

The ALU needed some modification from the prior phase. For this phase, the ALU contained a 4 bit select and two 32 bit inputs. A buffer, called F, of 32 bits was also used. The single output is called Compare, which is a single bit. The architecture for the ALU contains a process that involves the three inputs, s, A, and B. A case structure is set to determine what F’s value is. Afterwards, F is compared as well as the select to determine the value of Compare. We made an ALU control as well, which involved both a 6 bit input and a 2 bit input. The output is a 4 bit value called ALUOp, which is used in the architecture along with case statements to determine the value of the output. We used the megafunction for the data memory portion of this phase.

Other control units were also required for this phase. We have an entity called Control\_Main, which has a single input called inst, which is 6 bits. This is the most significant bites of the instruction. The outputs are RedDst, ALUSrc, MemtoReg, RegWrite, MemRead, MemWrite, Branch, and ALUControl. All of these outputs are 1 bit with the exception of ALUControl, which is 2 bits. Much like in the previous phases, we created our PC the exact same way, with a 32 bit input called D and two 1 bit inputs called CLR and CLK. The only output is Q, which is a 12 bit value. Q is determined in the architecture depending on the values of CLR and if there is a clock event. ShiftLeft2 is an entity that does just what it says – it shifts left two places. It includes a 32 bit input, A, and a 32 bit output, Q.

Pipelined Processor:

Everything included in the single cycle processor is also included in this phase of the project. The main difference is that pipeline registers were added. This included an IF/ID, ID/EX, EX/MEM, and MEM/WB pipeline register. The IF/ID register has three inputs, the 1 bit Clock and the two 32 bit inputs called PCAdd\_In and Instruction\_In. The two outputs called PCAdd\_Out and Instruction\_Out are both 32 bit values. Based on a clock event and Clock being equal to 1, PCAdd\_Out and Instruction\_Out are assigned. THE ID/EX register is rather large, with 14 inputs. Each of these inputs is related to the clock and each phase of the pipeline process. The 13 outputs also correspond to the phases of the pipelined process.

Forwarding and Hazard Unit Processor:

## Synthesis Results:

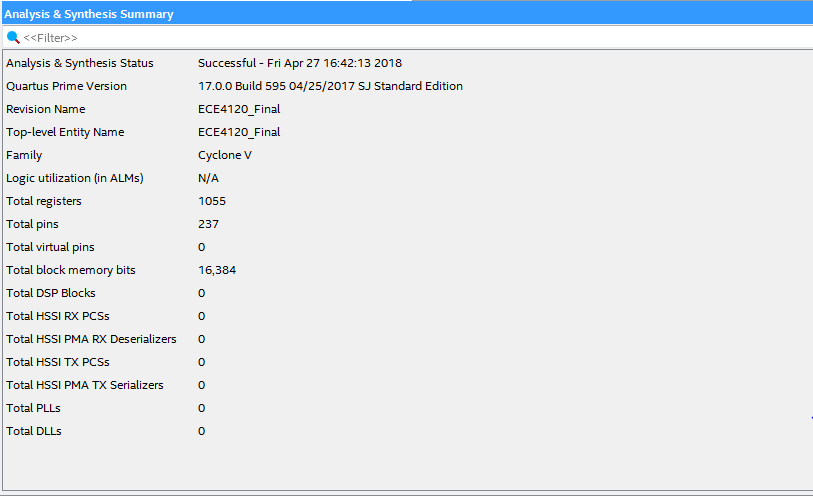


Figure 4. Single Cycle Synthesis Results

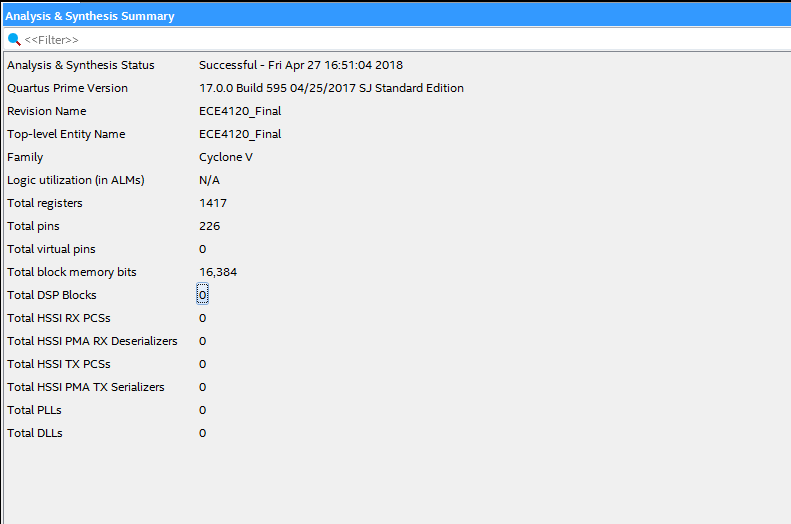


Figure 5. Pipelined Synthesis Results

Figure 6. Forward and Hazard Unit Synthesis Results

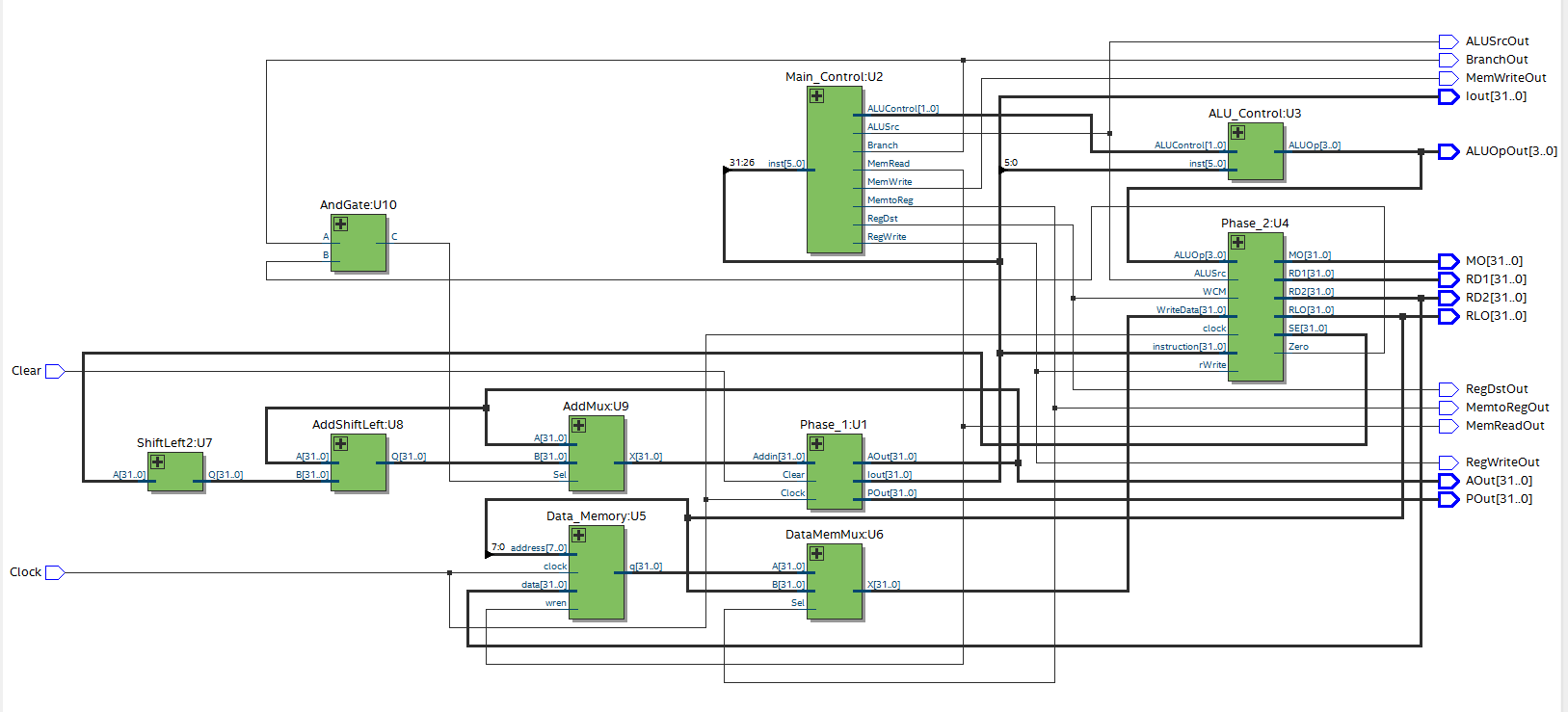


Figure 7. Single Cycle RTL Snapshot View

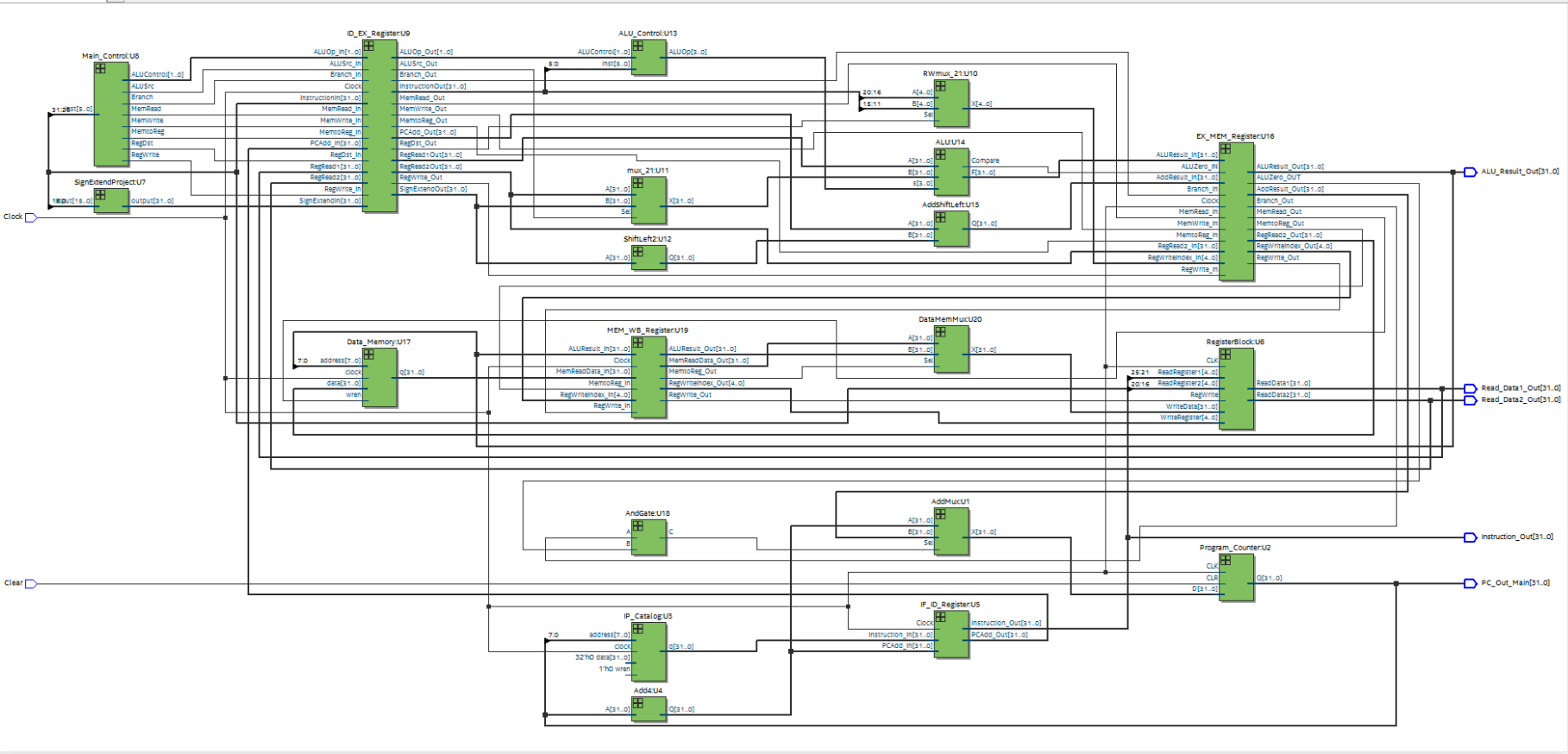


Figure 8. Pipelined Processor RTL Snapshot View

Figure 9. Forward/Hazard RTL Snapshot View

## Test Bench Elaboration:

Figure 10. Single Cycle Test Bench

Figure 11. Pipelined Test Bench

Figure 12. Forward/Hazard Test Bench

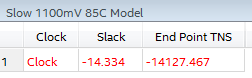
## Waveform Elaboration:

Figure 13. Single Cycle Waveform

Figure 14. Pipelined Waveform

Figure 15. Forward/Hazard Waveform

## TimeQuest Timing Analyzer:



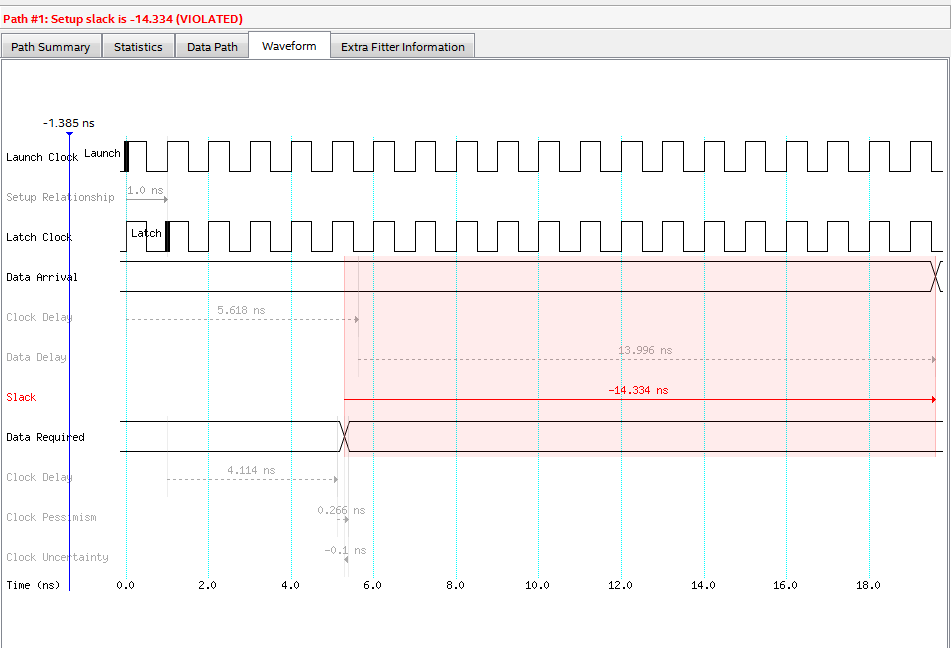
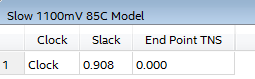


Figure 16. Single Cycle Setup Time



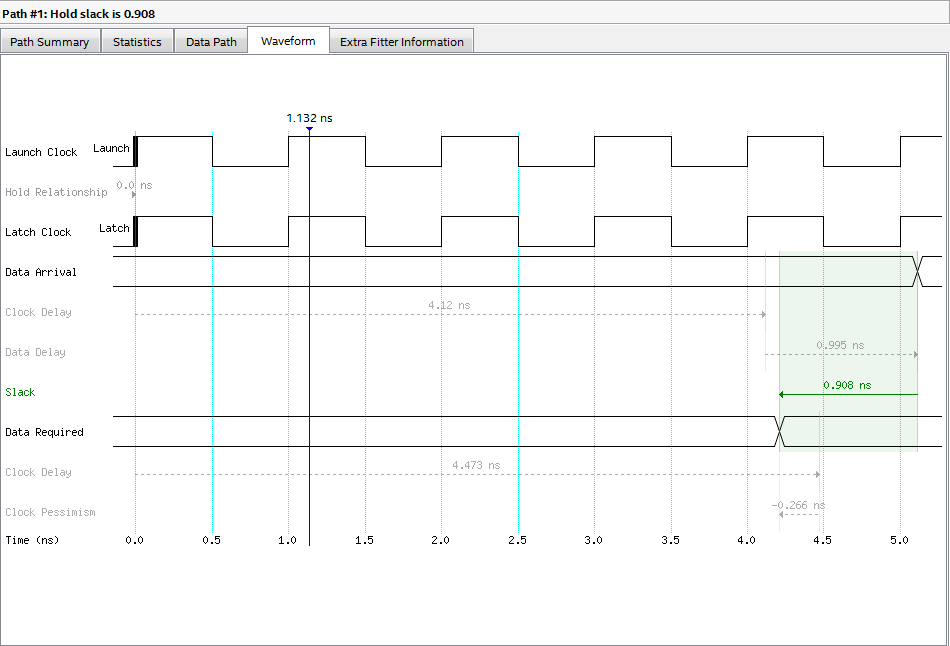
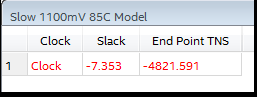


Figure 17. Single Cycle Hold Time



Figure 18. Single Cycle Fmax Report



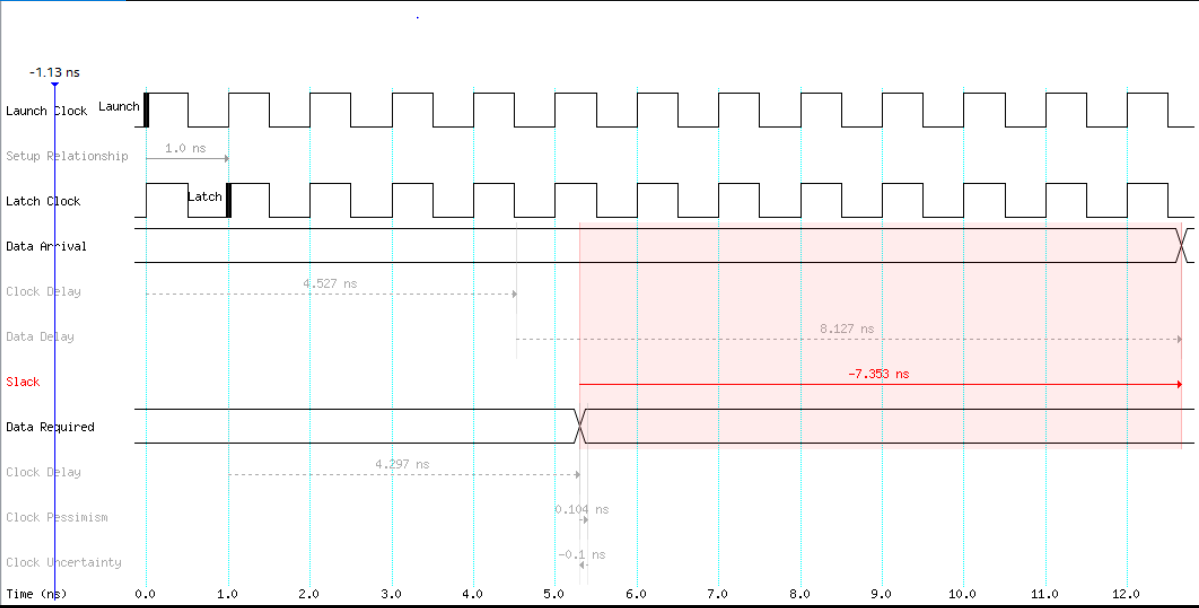
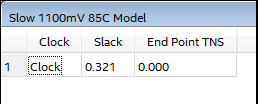


Figure 19. Pipelined Setup Time



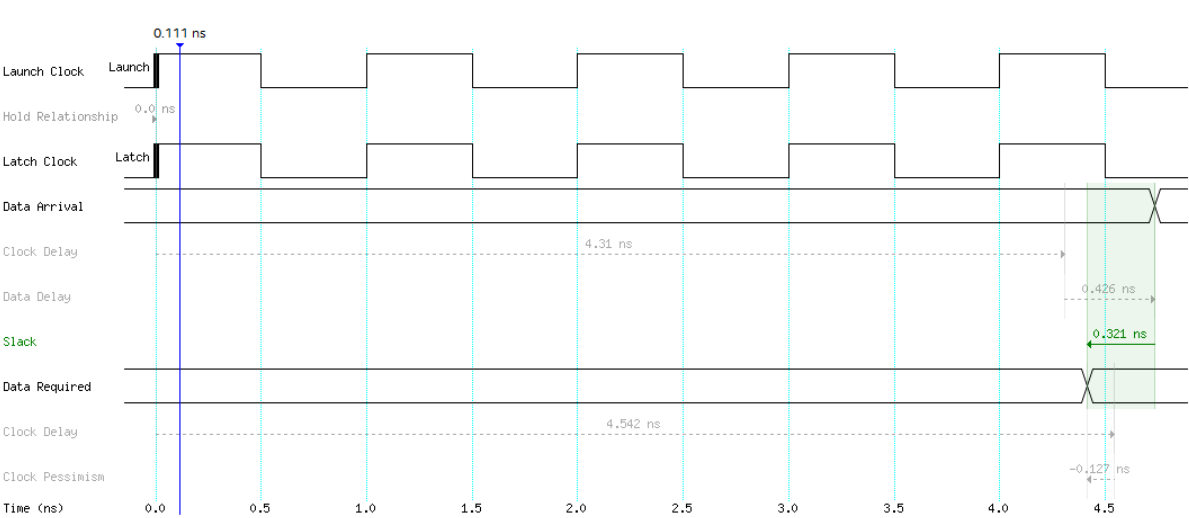


Figure 20. Pipelined Hold Time

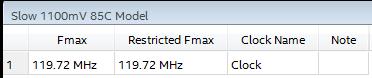


Figure 21. Fmax Report

## Phase Three Improvements:

The differences in performance/timing were pretty substantial overall. With the single cycle iteration allowing a maximum frequency of 65.21 Mhz and the pipelined iteration allowing 119.72 MHz. The delays (setup time, hold time) in the single cycle are reduced roughly by half in the pipelined implementation. While the pipeline registers have some additional individual setup time, it is insignificant in comparison to the setup time of the single cycle due to the reduced clock rate needed in order to run that many devices in one cycle while remaining stable.

## Pipeline Hardware Overhead:

## Forward and Hazard Units Performance Penalty:

## Forward and Hazard Units Hardware Overhead:

## Individual Contribution:

All of the members of the group met multiple times to complete phase three; however, we did have certain people designated to specific tasks. Hunter decided to tackle the ALU modifications and pipelining. Brantlee worked on the control unit, the hazard unit, and TimeQuest. Josh focused on the forwarding unit. Cameron worked on the control unit, the additional data memory unit, and all the smaller components required for each unit to receive the correct data. Randy worked on the hazard unit, TimeQuest, and wrote the report.

## Phase Three Conclusion:

Phase three was much more involved than any of the other phases. With so much to keep track of, things quickly got difficult. Despite the challenge, our group was able to obtain a correct design for phase three, implementing both Figure 1 and Figure 2 from the project description given. The additions of branch handling hardware via modifying the ALU, a memory unit for data memory, the auxiliary multiplexers all were designed and implemented correctly to achieve the first step of phase three. Afterwards, pipelining being implemented as the next step. It was a very challenging phase of the project; however, it was a major learning experience for the entire group.