Luis Ruiz

5001441817

CPE 300-1001

Final Project

1. The project required us to build a microprocessor. The microprocessor made us apply our knowledge on what we have been learning these past few months and build off of it. The processor itself required us to build a data path and control unit that would decode and execute a set of instruction. In essence the processor is a complier which grabs a certain amount of instructions and executes the instructions by decoding it using a control unit, the control unit then gives the data path the bits in order to execute the instruction.

   Most of the instructions in my control unit require 3 cycles and only a few require 4 cycles. The first cycle is always the Fetch cycle, the 2nd cycle is dedicated to decoding what instruction is to be executed, and 3rd cycle is usually the cycle where the instruction is executed. A 4th cycle is required for instructions such as LDM and STM where a value is being stored into my Accumulator and or the Memory itself.

Instructions to Be Executed

000000: LDI A, 5 // Accumulator <- 5

000010: DEC A //Accumulator <-4

000011: JNZ 000010 //PC <-000010 if Accumulator! =0

000101: INC A //Accumulator<-1

000110: STM 001111, A //RAM [001111] <-1

001000: STA 0, A //RF [000] <-1

001001: ADD A, 0 //A <- RF [000] +1

001010: STM 010000, A //RAM [010000] <- 2

001100: HALT //STOP

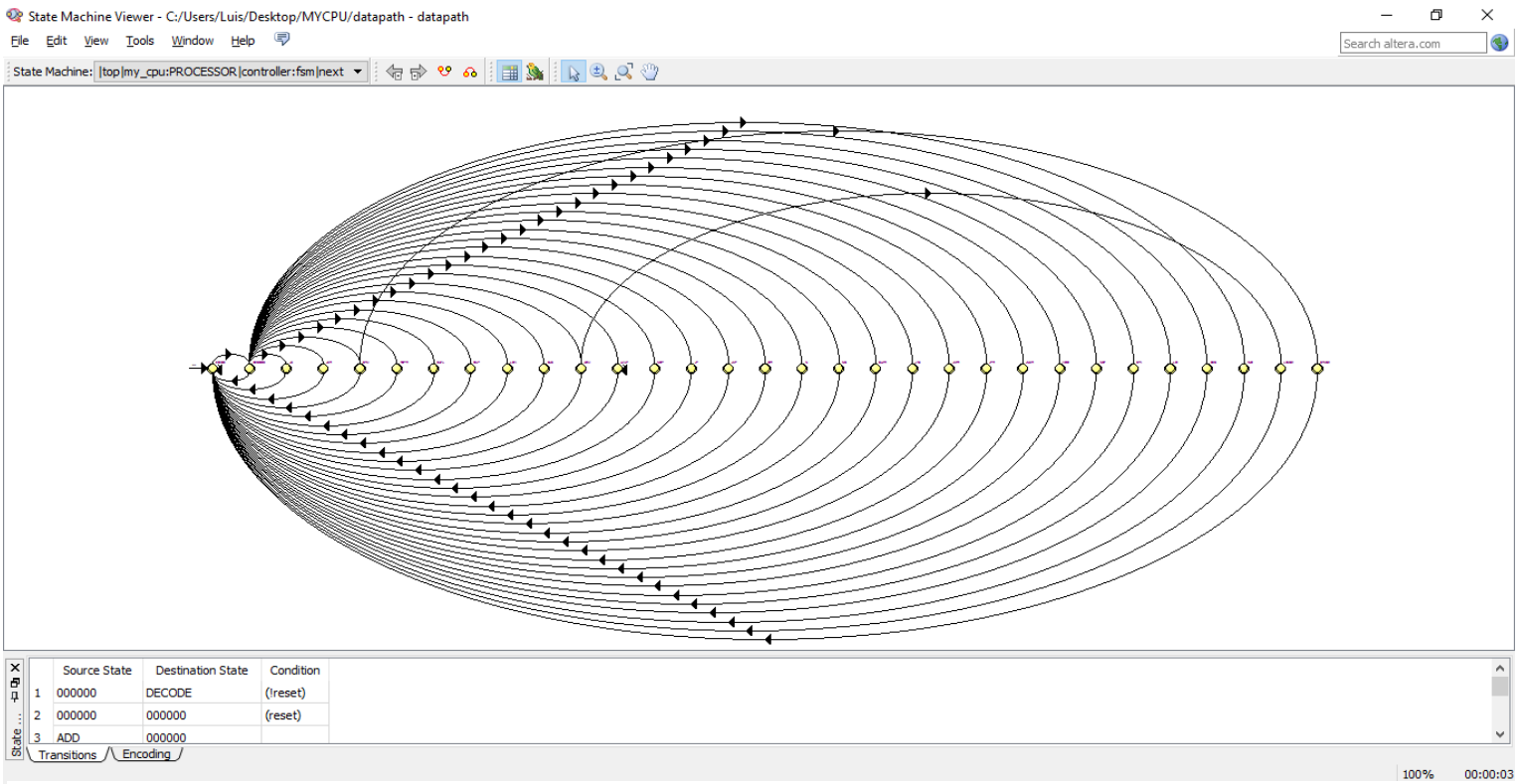
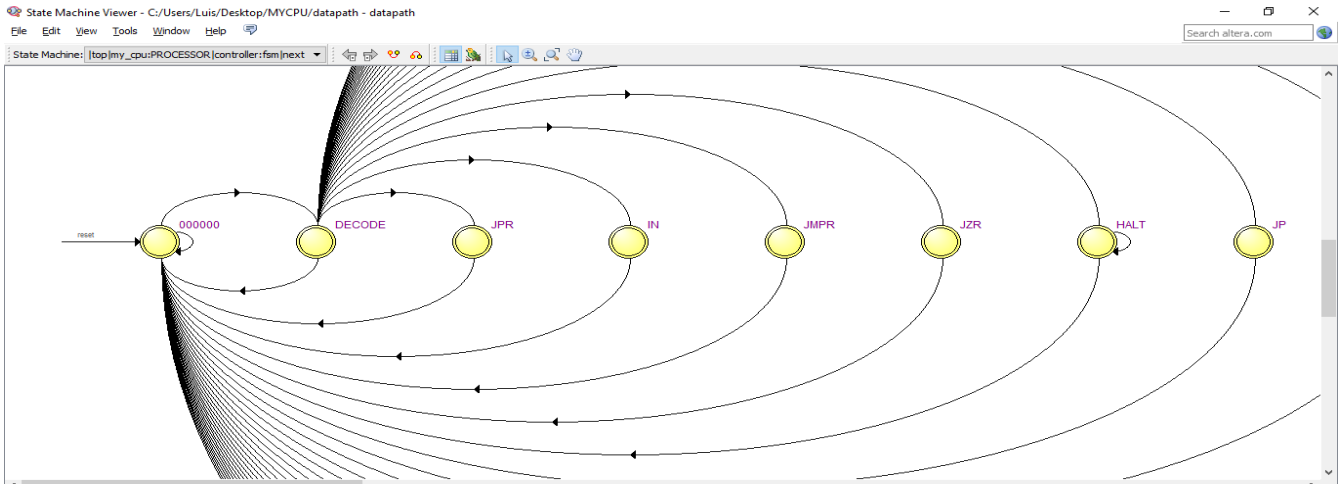


Figure 1: FSM Schematic



2. Describe the control unit design.
   a) How many instructions are decoded and the ALUSel and ShiftSel encoded?
      i. The number of instructions that were decoded were 27 distinct instructions. Each instruction was dedicated a specific state and the control signals were given at the present state.
   b) Whether the control circuit is FSM or Clock counter+ decoder+ Combination
      i. 29 states, one for fetch and decode and then 27 distinct states, one for each instruction. When the code started the instructions were initially stored into my Memory. Once stored into Memory the code was fetched on the first clock pulse and then decoded. Once it was decoded our present state was the instruction and this is where the controls to the data path are given.

3. My Hierarchy Module is named top. v; Top basically instantiates the memory and a module called my_cpu which contains the data path and control unit.
4. Screen Shots are of the wave form are as follows:

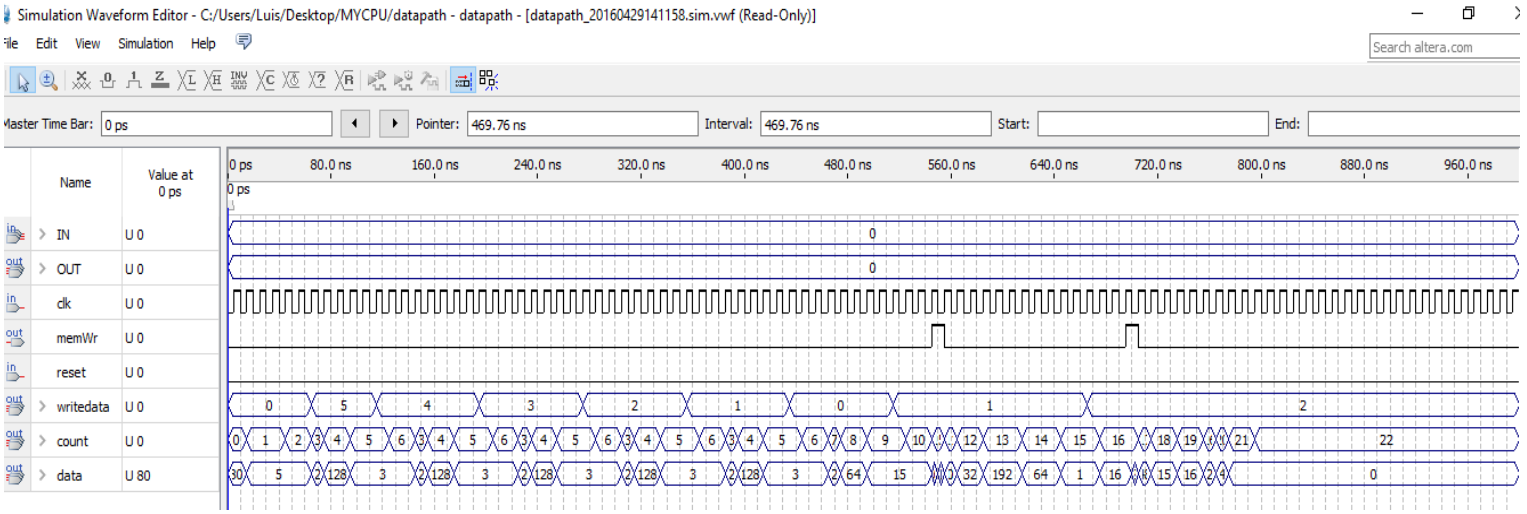


Figure 2: Wave Form, Writedata is the content in the Accumalator, Count was the PC output, and data are the instruction being outputed by my Memory

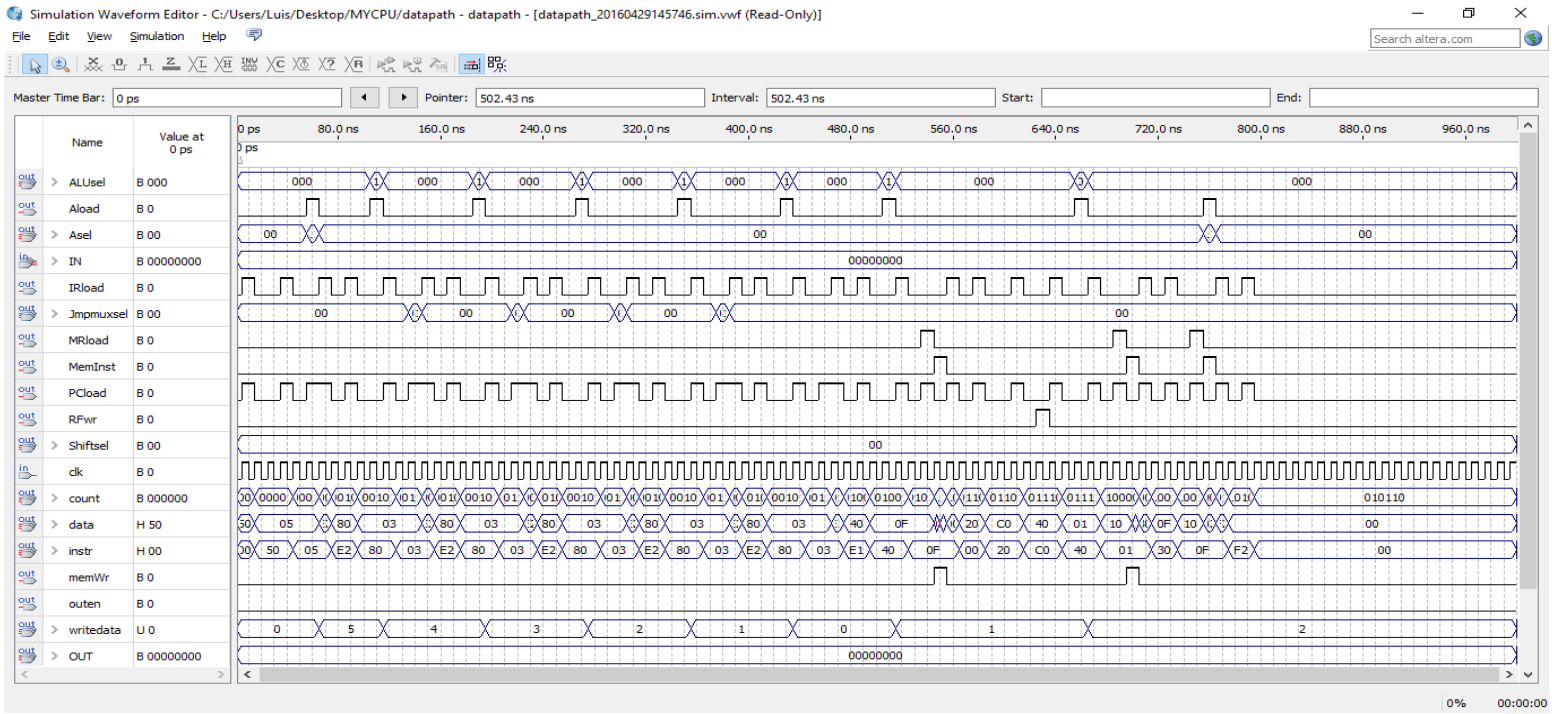


Figure 3: More Descriptive

5. The Test bench I created really only generates a clock pulse and ends when all my instructions have been fully executed. The Test bench display a message saying it has finished when it has all the expectations have been met. The screen shots provided will display the values that were stored in the memory location 001111, 010000, and also the content of what R [0] holds.

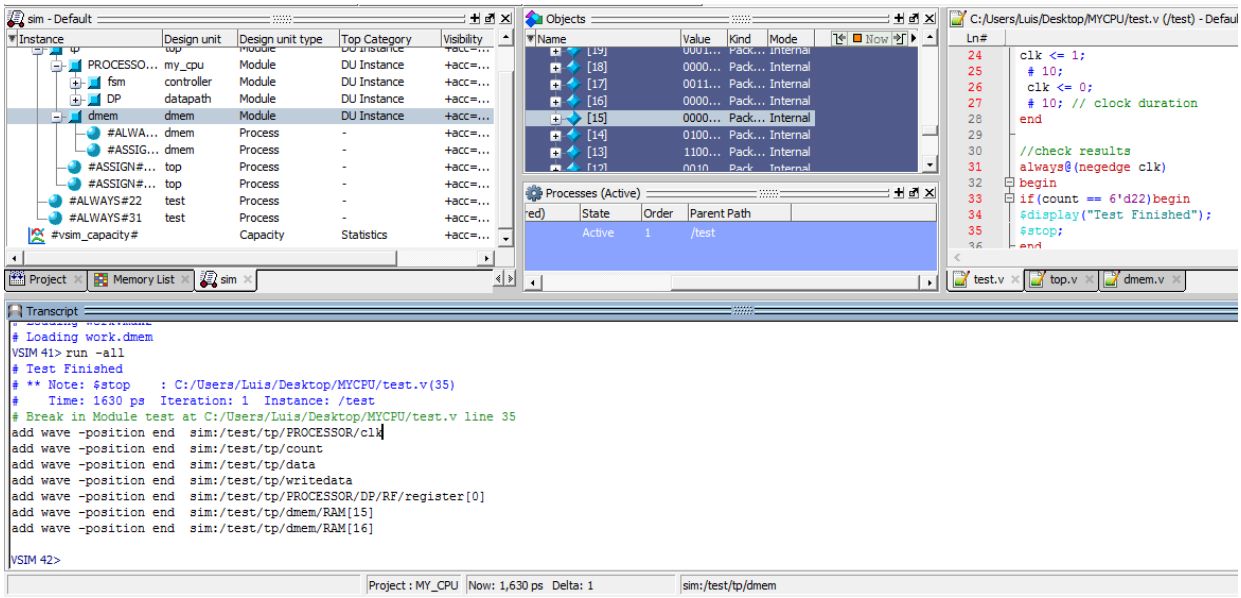


Figure 4: Shows the instruction that were added to display on the wave form

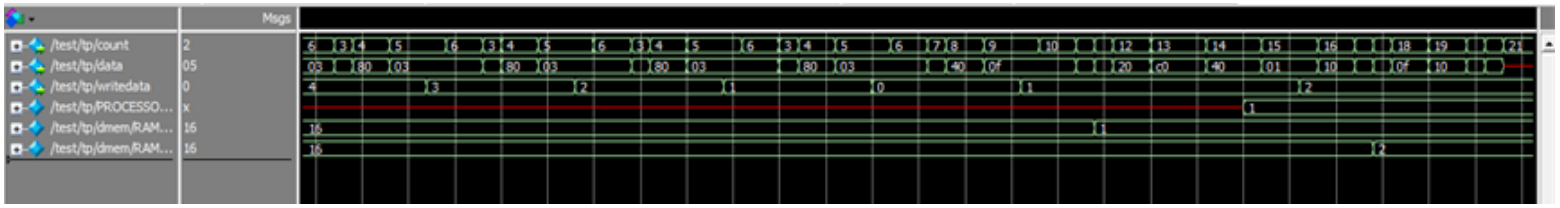


Figure 5: Shows the wave form, from model slim

6. The clock requires a period 8.50ns in order to have a positive slack of 0.113ns. This means that the longest instruction took roughly about 8.35 ns in order to complete thus giving us a positive slack.

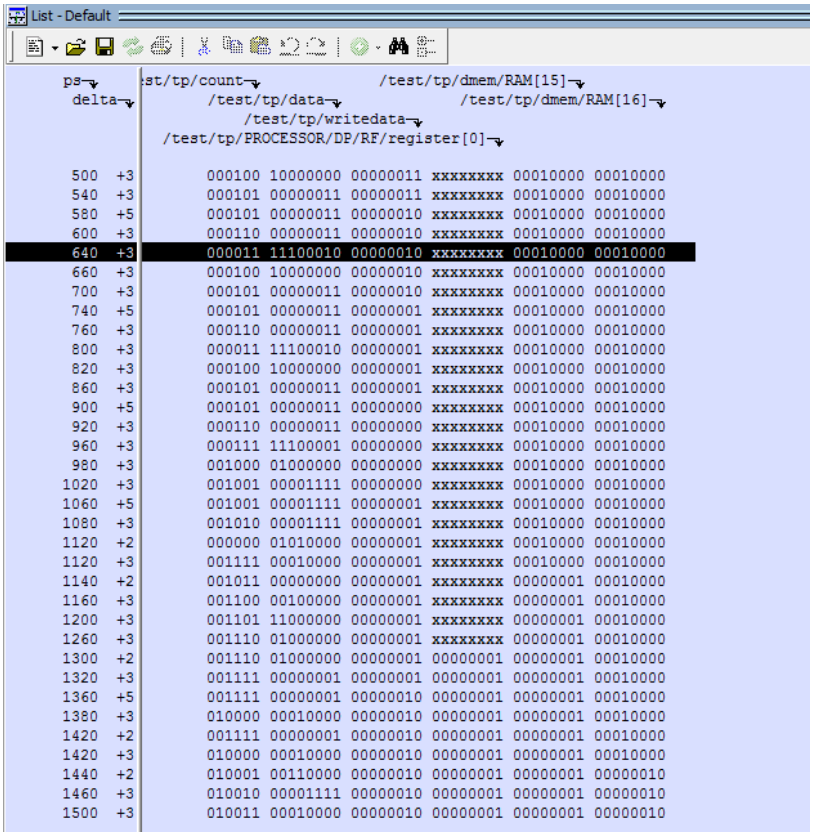


Figure 6: List which also displays R[0] Mem 010000,001111

add list -width 16 /test/tp/count

add list /test/tp/data

add list /test/tp/writedata

add list {/test/tp/PROCESSOR/DP/RF/register[0]}

add list {/test/tp/dmem/RAM[15]}

add list {/test/tp/dmem/RAM[16]}

configure list -usestrobe 0

configure list -strobestart {0 ps} -strobeperiod {0 ps}

configure list -usesignaltrigger 1

configure list -delta all

configure list -signalnamewidth 0

configure list -datasetprefix 0

configure list -namelimit 5

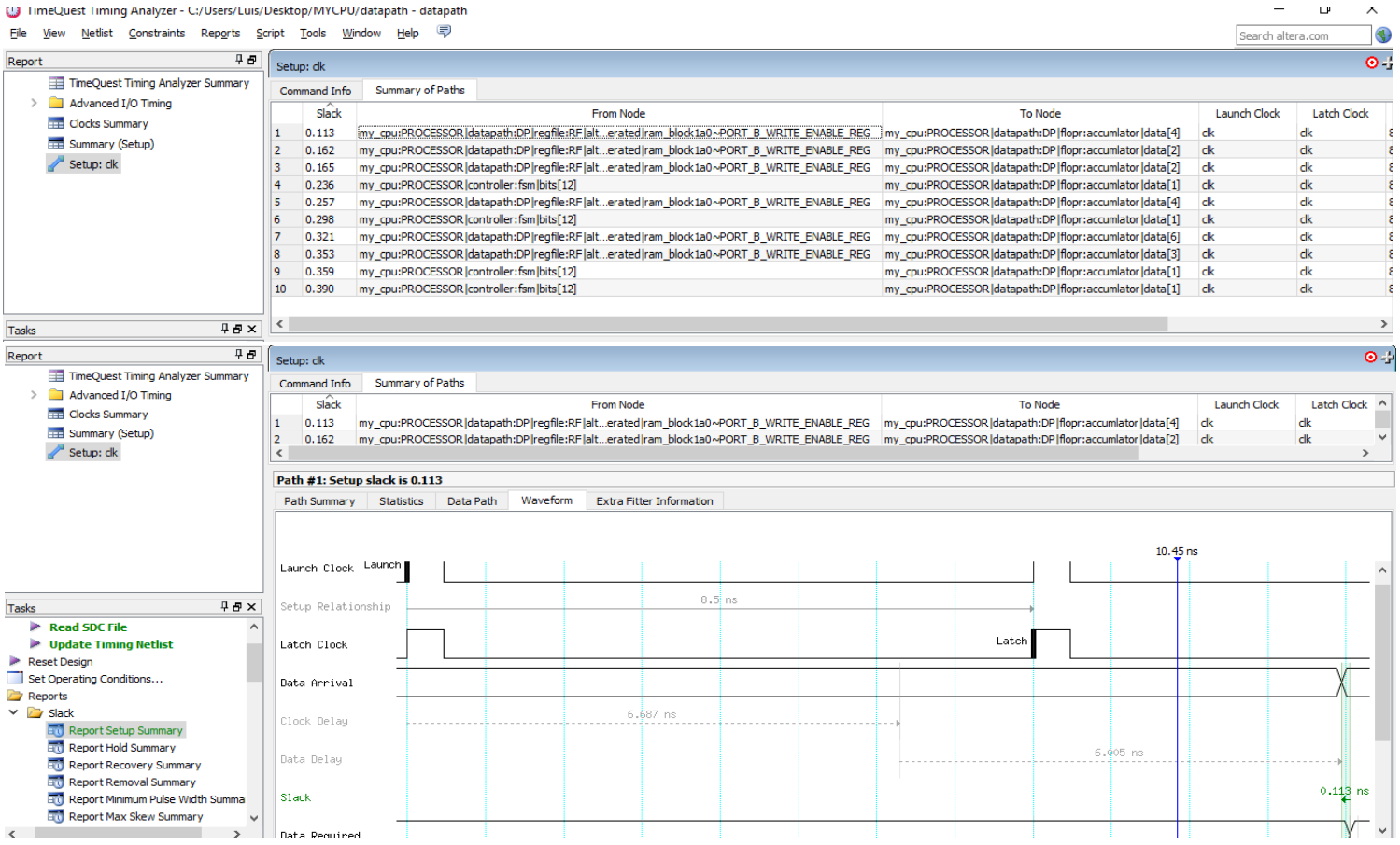


Figure 7: Time Quest Analysis

7. The only hiccup I had throughout the project were instructions that had two bytes. For some reason I have to place the 2nd bytes twice in order for it to work you will notice it in my memfile.dat. My guess was it required a delay in order to keep the 2nd byte longer.