Lab 04
Jonathan Gusler
2/11/2019

Overview

This lab involved finding the clock rate limit of a simplified MIPS RISC processor, in other words, its ability to produce correct results for a given program. This involved running the program multiple times at increasing clock rates by adjusting the clock period until incorrect results started occurring. The level of accuracy sought for the period limit was only to the nanosecond.

Background

This lab used the exact same processor and instructions used in Lab 03 [1]. This lab requires knowledge of digital logic timing limitations, familiarity with Quartus, MIPS, and machine code. The code for this lab comes from Section 14 in *Rapid Prototyping of Digital Systems, SOPC Edition,* by Hamblen, Hall, and Furman [2].

Discussion

After compiling the code and ensuring the code performed as intended, multiple timing analyses were run. The clock period was adjusted each time using a binary search method to find the shortest period the clock could operate at before errors occurred. The first simulation had a clock period of 100 ns, Figure 1. Since no errors were produced at this speed, the next simulation was run at 50 ns, Figure 2, then 25 ns. At 50 ns no errors occurred. At 25 ns errors were present, so the period was increased to 37.5 ns, Figure 3. This process continued until settling upon a bare minimum required period of 38 ns, Figure 4, equivalent to a clock rate of 26.32 MHz.

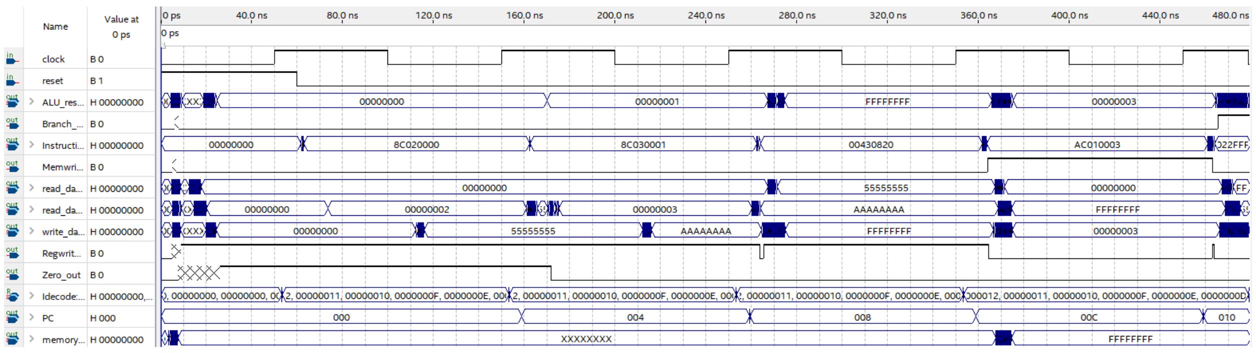


Figure 1: 100 ns period simulation.

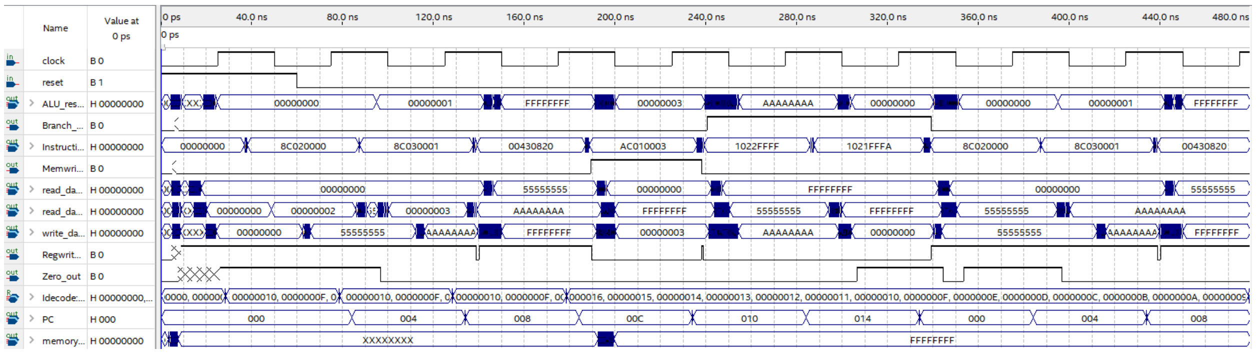
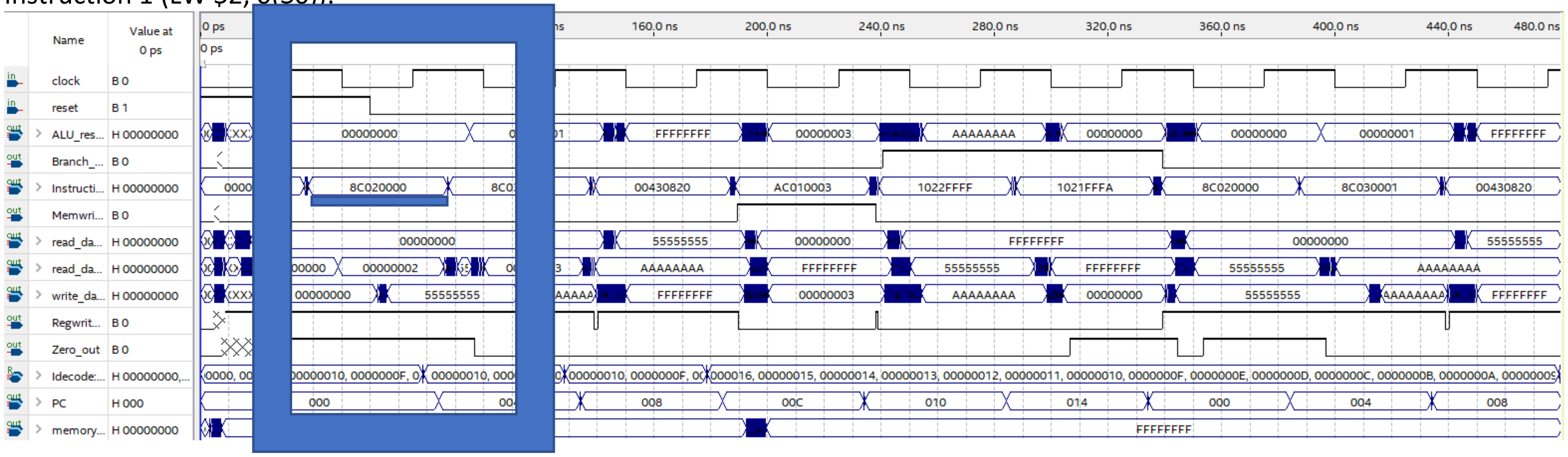


Figure 2: 50 ns period simulation.

For the 50 ns simulation, the instruction memory contained instructions for load and store word, add, and branch on equals. The initial data memory value is undefined, the final is 0xFFFFFFFF. An analysis of the signals, based on the instructions which are one cycle long, starting at the first instruction of 8C020000, for the 50 ns simulations begins on the next page (all values are in hexadecimal and each waveform is the 50 ns simulation):

Instruction 1 (LW $2, 0($0)):



PC and Branch calculation:

PC: 000

Branch Calculation: No branch

Register file read/write data:

Register Data Write: 55555555

Read Data 1: 00000000

Read Data 2: 00000002

ALU input and output:

Input: 00000000 and 00000000

Output: 00000000

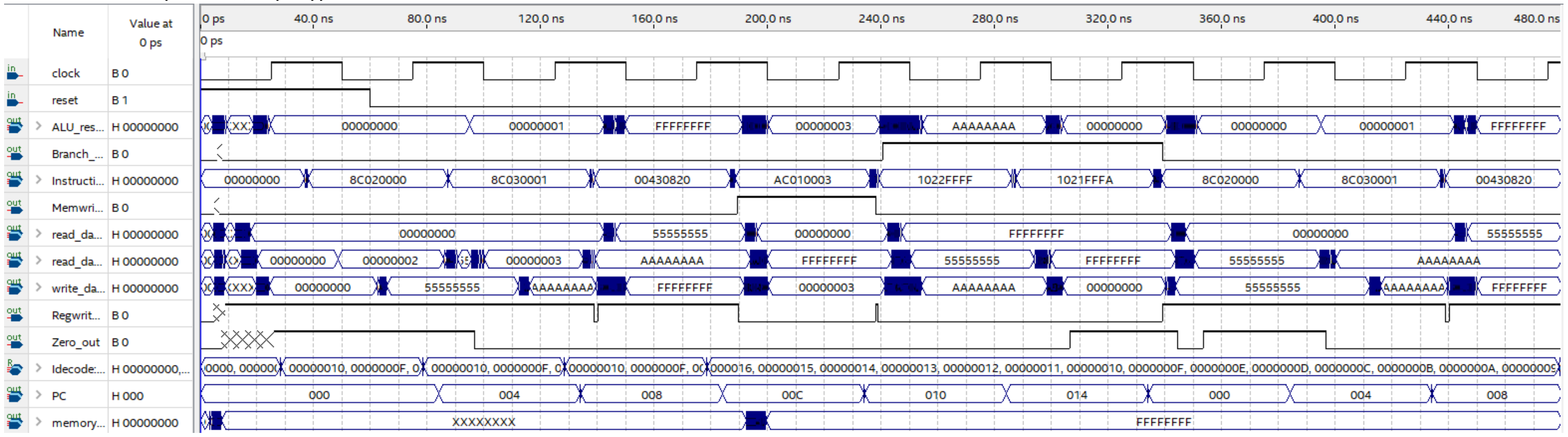
Data memory address, data in/out:

Address: 0($0)

In: 00000000

Out: 5555555

Instruction 1 (LW $2, 0($0)):

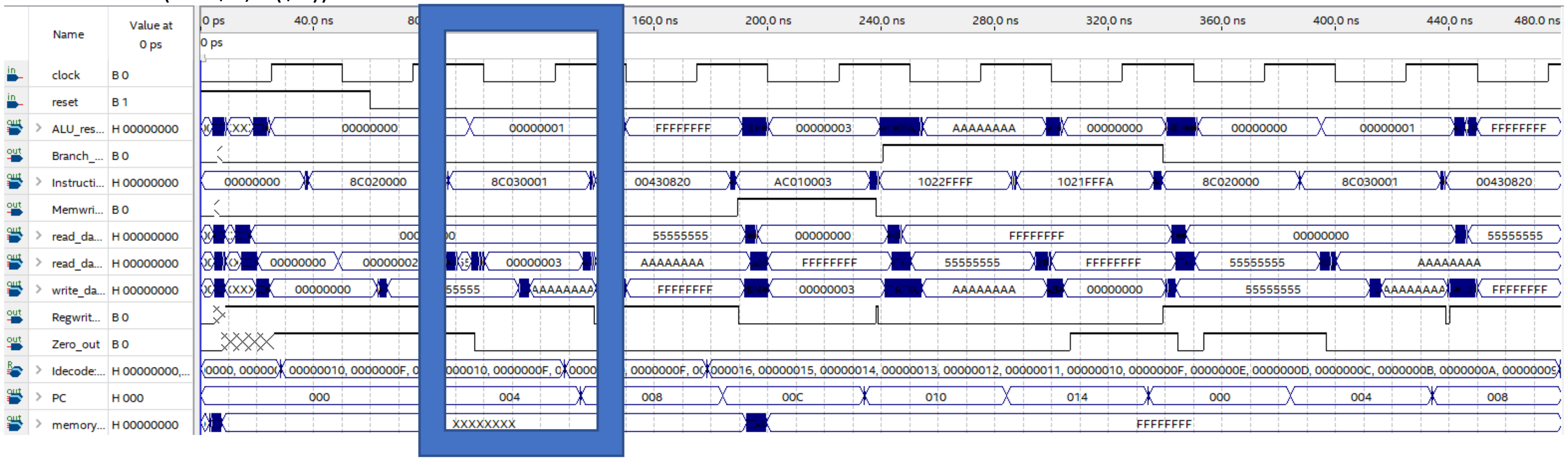


Name	Value at 0 ps
clock	B 0
reset	B 1
ALU_res...	H 00000000
Branch_...	B 0
Instructi...	H 00000000
Memwri...	B 0
read_da...	H 00000000
read_da...	H 00000000
write_da...	H 00000000
Regwrit...	B 0
Zero_out	B 0
Idecode:...	H 00000000,...
PC	H 000
memory...	H 00000000

Control lines asserted:

- RegWrite
- MemRead
- MemtoReg
- RegDst
- ALUSrc
- ALUOp

Instruction 2 (LW $3, 1($0)):



PC and Branch calculation:

PC: 004

Branch Calculation: No branch

Register file read/write data:

Register Data Write: AAAAAAAA

Read Data 1: 00000000

Read Data 2: 00000003

ALU input and output:

Input: 00000000 and 00000001

Output: 00000001

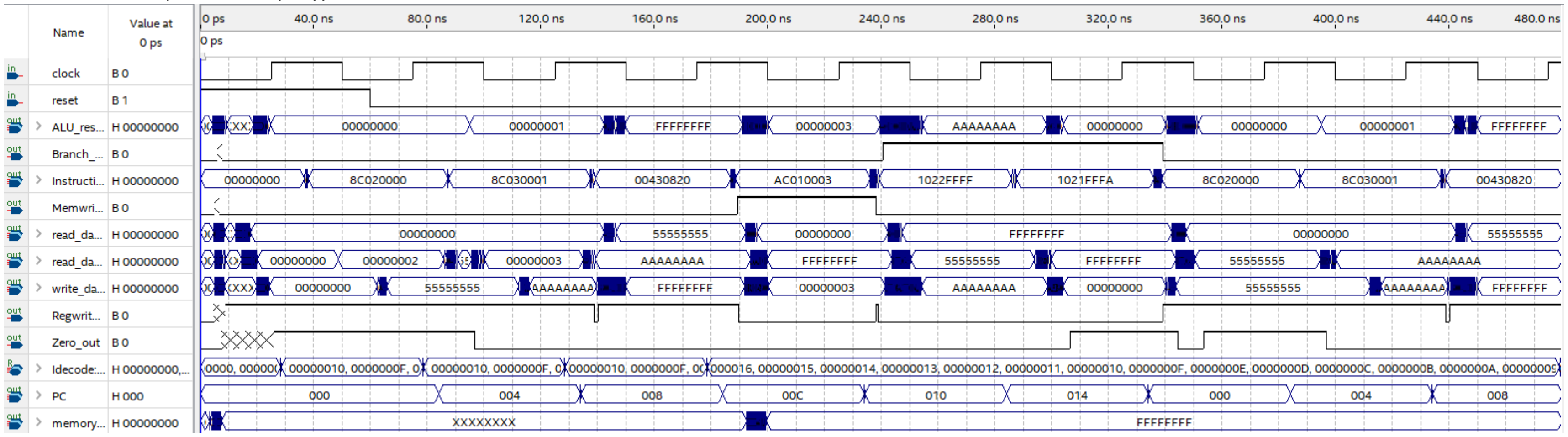
Data memory address, data in/out:

Address: 1($0)

In: 00000001

Out: AAAAAAAA

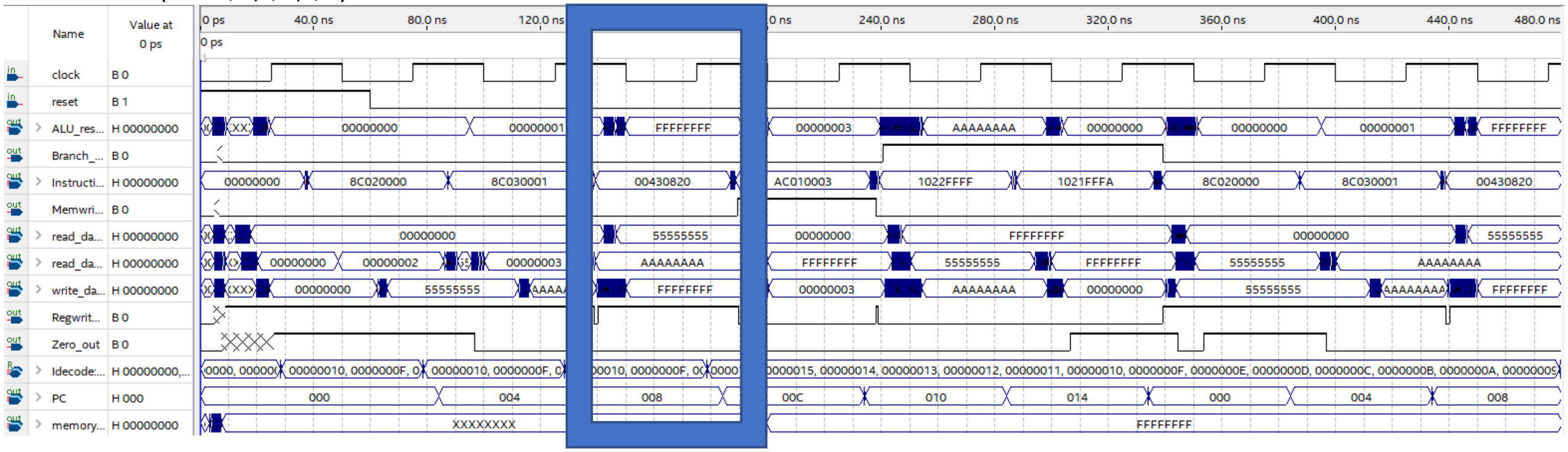
Instruction 2 (LW $3, 1($0)):



Control lines asserted:

- RegWrite
- MemRead
- MemtoReg
- RegDst
- ALUSrc
- ALUOp

Instruction 3 (ADD $1, $2, $3):



PC and Branch calculation:

PC: 008

Branch Calculation: No branch

Register file read/write data:

Register Data Write: FFFFFFFF

Read Data 1: 55555555

Read Data 2: AAAAAAAA

ALU input and output:

Input: AAAAAAAA and 55555555

Output: FFFFFFFF

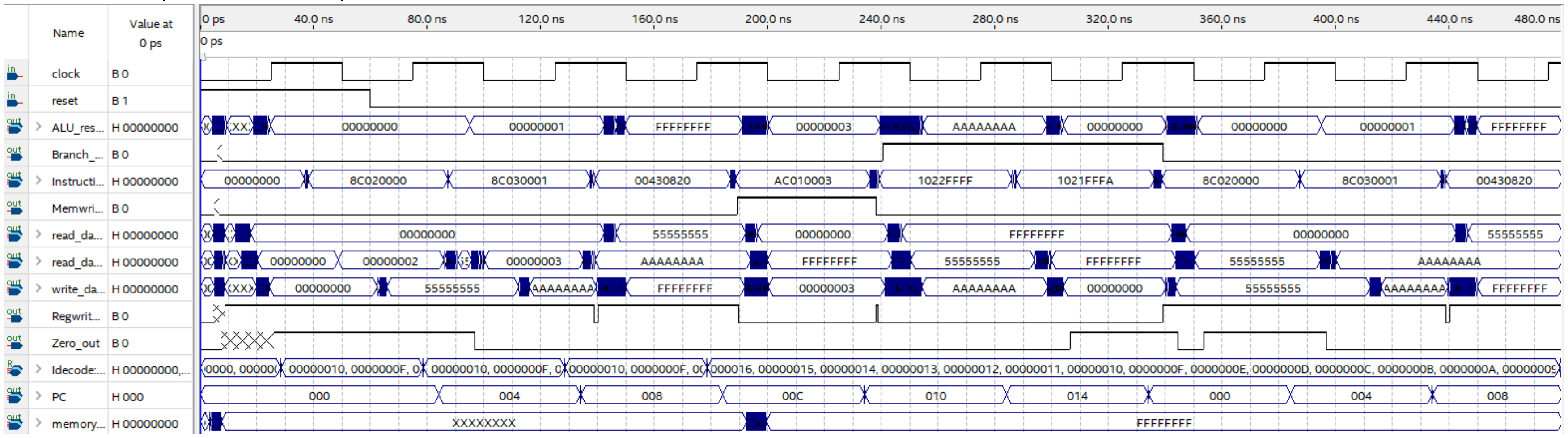
Data memory address, data in/out:

Address: 1($0)

In: N/A

Out: N/A

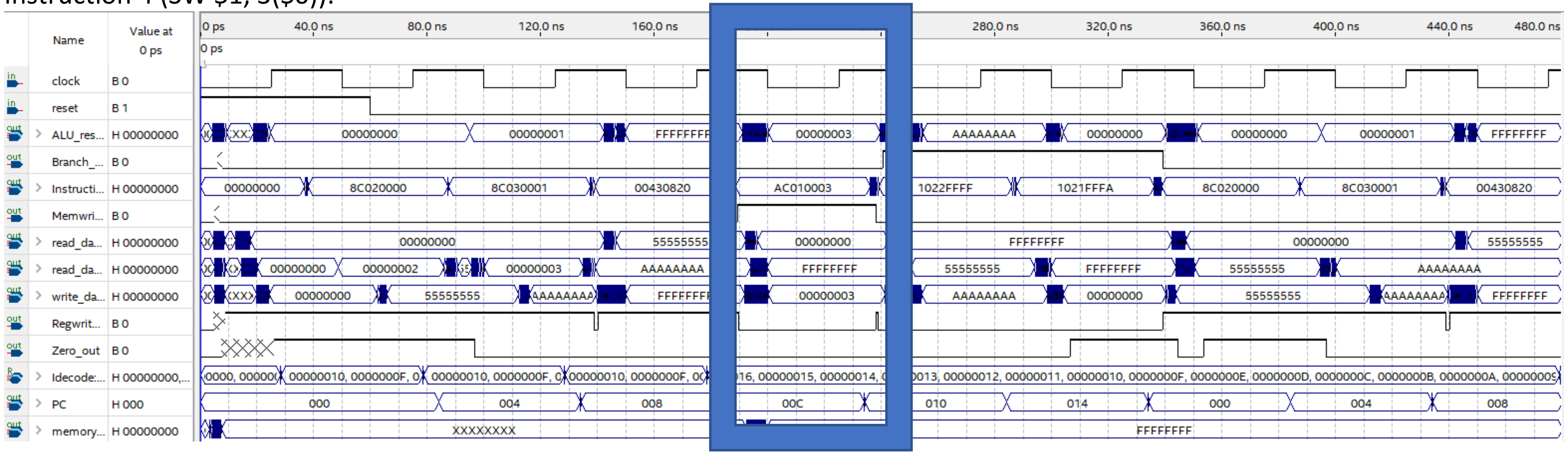
Instruction 3 (ADD $1, $2, $3):



Control lines involved:

- RegWrite
- RegDst
- ALUOp

Instruction 4 (SW $1, 3($0)):



PC and Branch calculation:

PC: 008

Branch Calculation: No branch

Register file read/write data:

Register Data Write: FFFFFFFF

Read Data 1: 55555555

Read Data 2: AAAAAAAA

ALU input and output:

Input: AAAAAAAA and 55555555

Output: FFFFFFFF

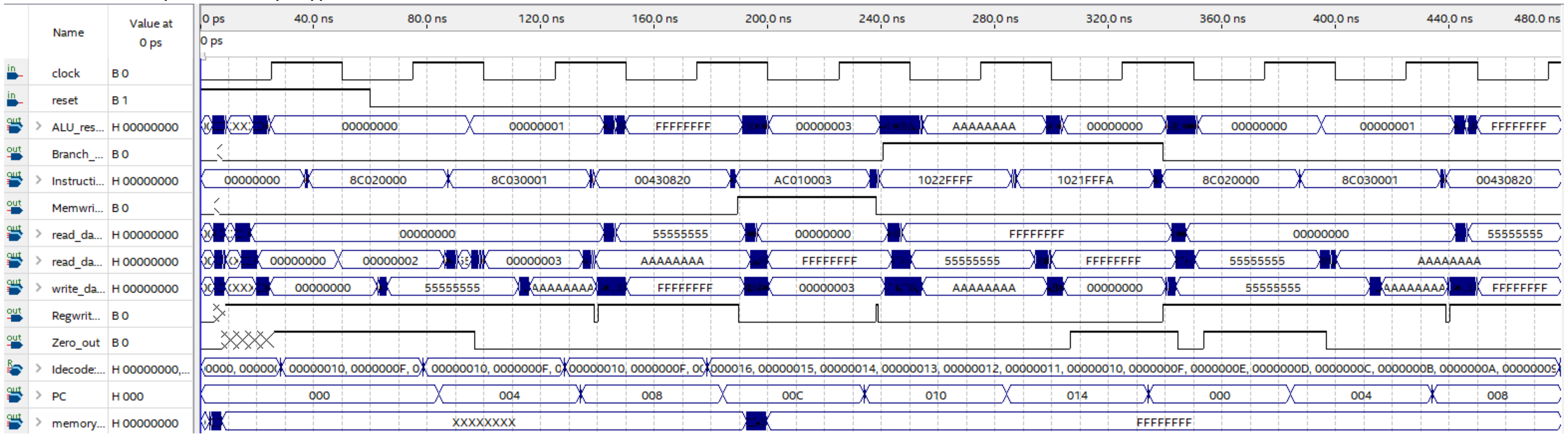
Data memory address, data in/out:

Address: 1($0)

In: N/A

Out: N/A

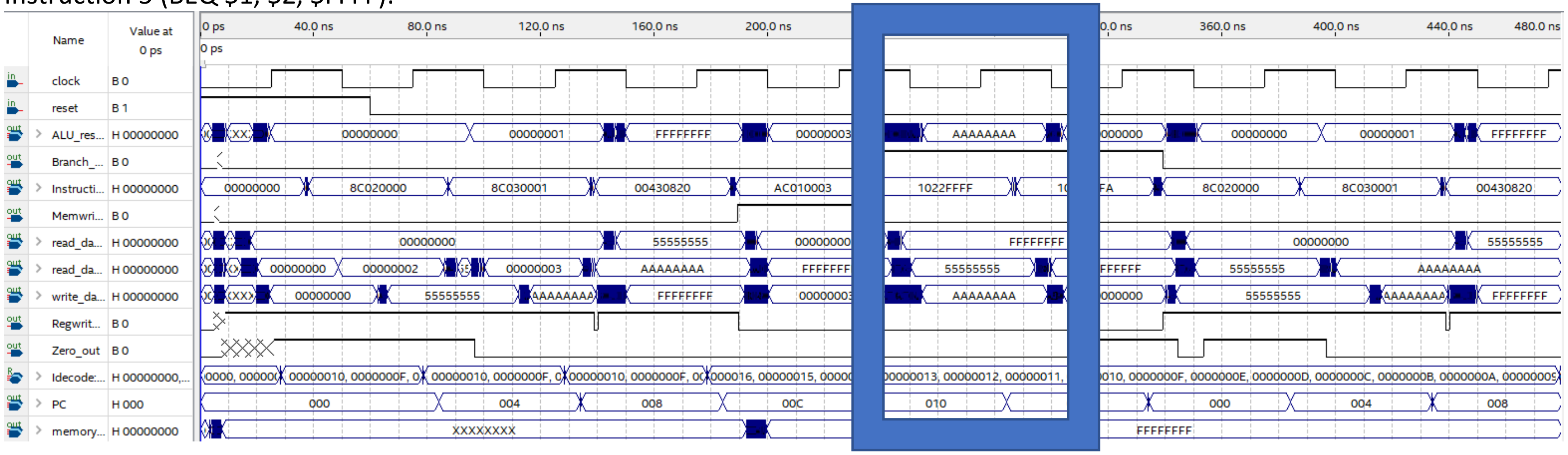
Instruction 4 (SW $1, 3($0)):



Control lines asserted:

- ALUOp
- MemWrite

Instruction 5 (BEQ $1, $2, $FFFF):



PC and Branch calculation:

PC: 010

Branch Calculation: No branch

Register file read/write data:

Register Data Write: N/A

Read Data 1: FFFFFFFF

Read Data 2: 55555555

ALU input and output:

Input: FFFFFFFF and 55555555

Output: AAAAAAAA

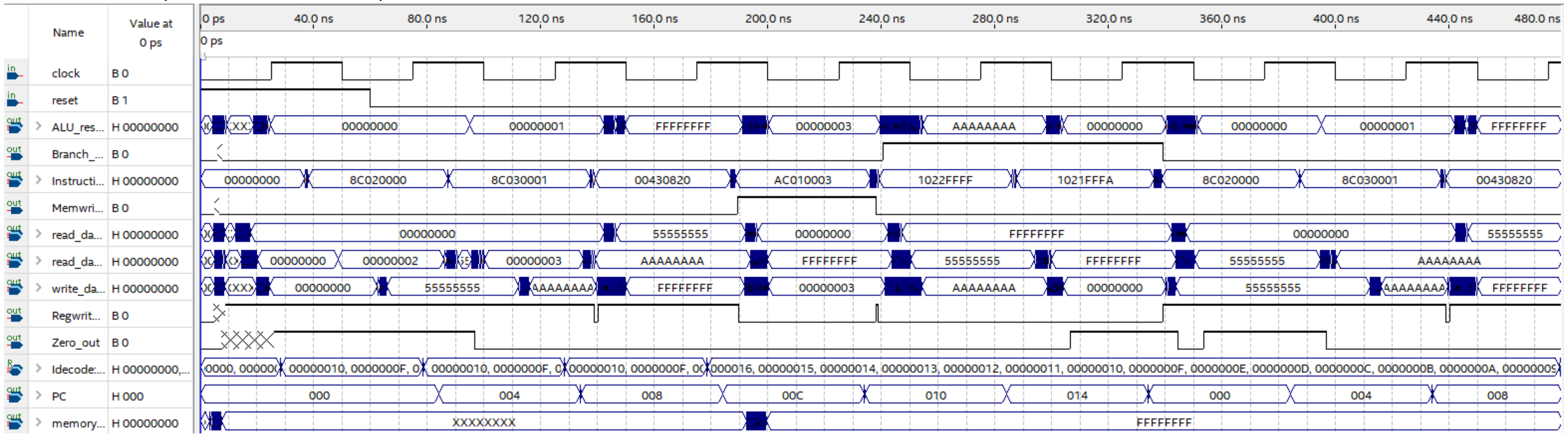
Data memory address, data in/out:

Address: N/A

In: N/A

Out: N/A

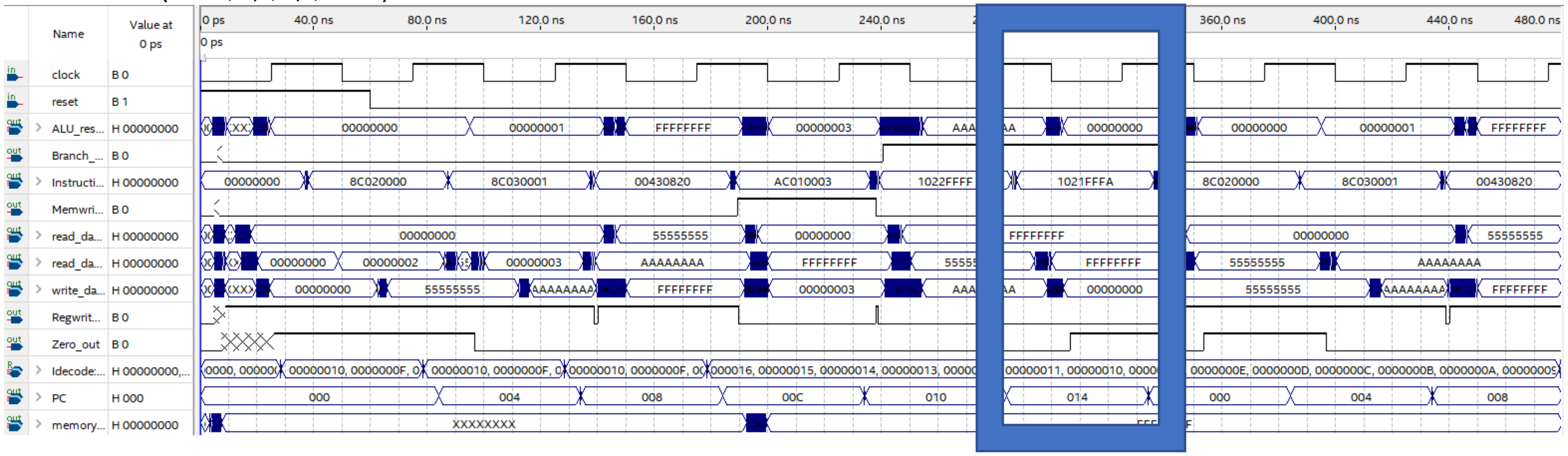
Instruction 5 (BEQ $1, $2, $FFFF):



Control lines asserted:

- ALUOp
- ALUSrc

Instruction 6 (BEQ $1, $1, $FFFA):



PC and Branch calculation:

PC: 0014

Branch Calculation: Branch

Register file read/write data:

Register Data Write: N/A

Read Data 1: FFFFFFFF

Read Data 2: FFFFFFFF

ALU input and output:

Input: FFFFFFFF and FFFFFFFF

Output: 00000000

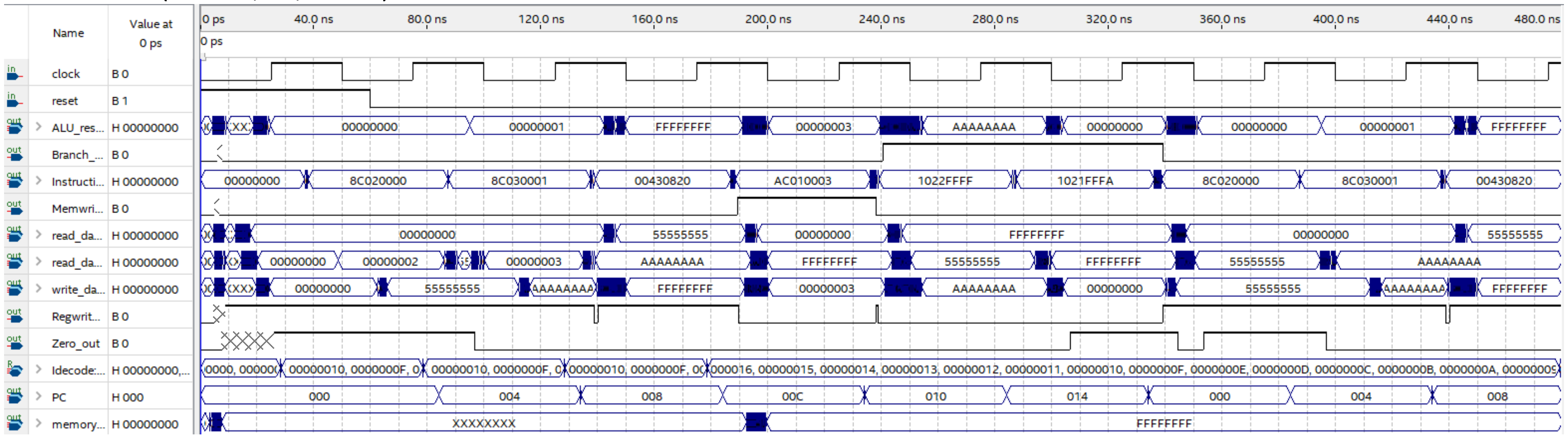
Data memory address, data in/out:

Address: N/A

In: N/A

Out: N/A

Instruction 6 (BEQ $1, $1, $FFFA):



Control lines involved:

- ALUSrc
- ALUOp
- PCSrc

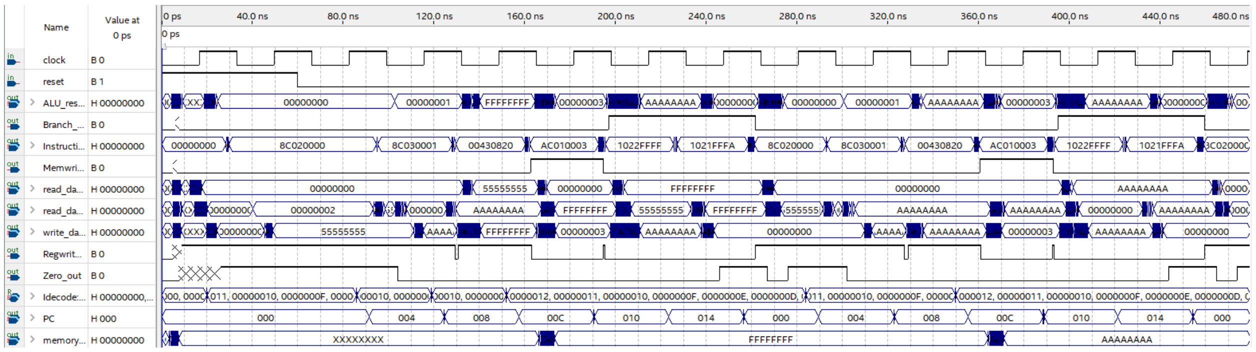


Figure 3: 37.5 ns period simulation.

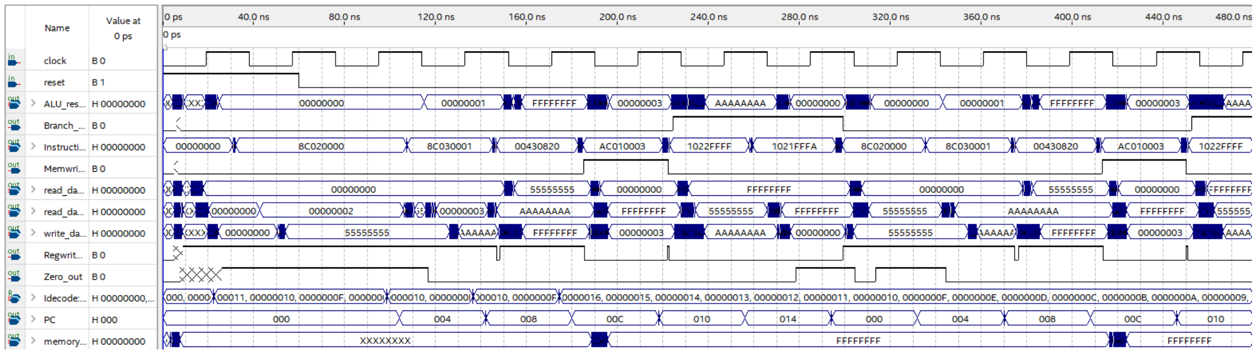


Figure 4: 38ns period simulation.

Analysis and Results

26 MHz is the clock rate limit. This value comes from taking the inverse of 38 ns, which is the period limit. Once the period drops lower than 38 ns, values that had not previously shown up in any simulation start appearing, leading to the conclusion that they are errors, which is backed by those different results not being the correct results of the assembly code.

Conclusion

This lab has been a good source of tangible evidence of the effects of violating set-up and hold times in digital logic timing and, subsequently, the limits of single cycle processors. A proposed solution to these limits is to begin executing the next instruction while the current one is running, like an assembly line. This design comes with its own rules that must be followed which will be studied at a later time.

References

[1] J. Gusler, "Lab 03," 2019.

[2] H. F. Hamblen, in *Rapid Prototyping of Digital Systems, SOPC Edition*, p. Section 14.