Tyler Kim

Digital Logic Design

Control Signal Logic

Component

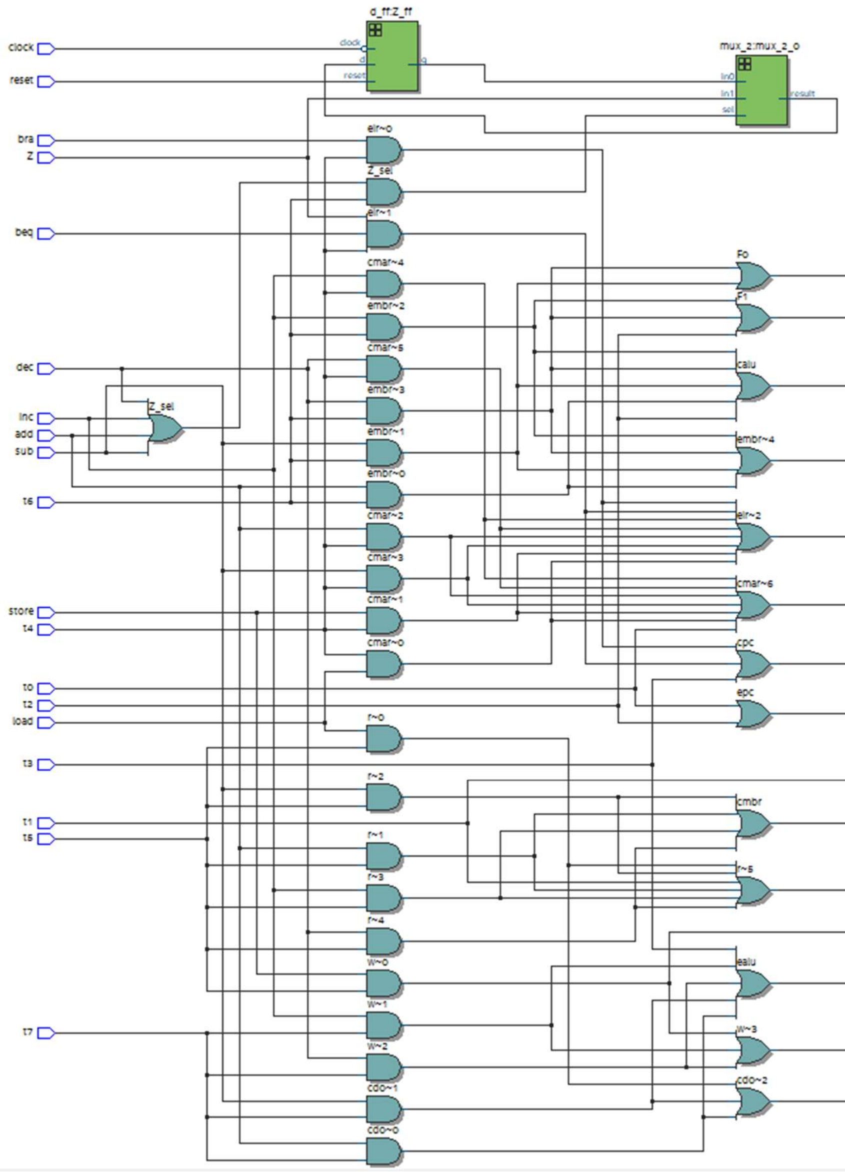
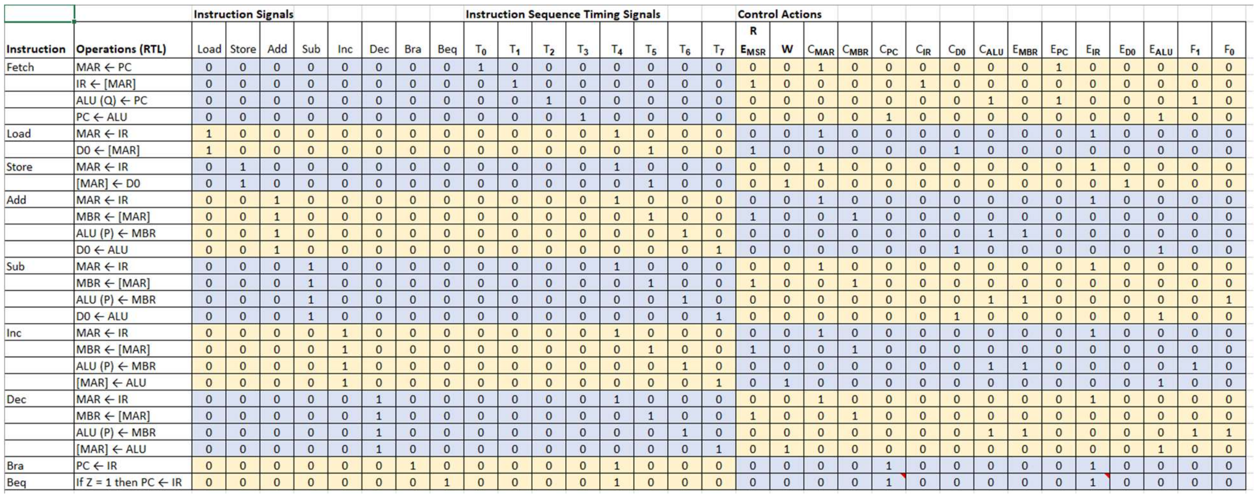


Figure 1: Control Signal Logic Schematic



		Instruction Signals								Instruction Sequence Timing Signals								Control Actions														
Instruction	Operations (RTL)	Load	Store	Add	Sub	Inc	Dec	Bra	Beq	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	R E_{MSR}	W	C_{MAR}	C_{MBR}	C_{PC}	C_{IR}	C_{D0}	C_{ALU}	E_{MBR}	E_{PC}	E_{IR}	E_{D0}	E_{ALU}	F_1	F_0
Fetch	MAR ← PC	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
	IR ← [MAR]	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	ALU (Q) ← PC	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0
	PC ← ALU	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
Load	MAR ← IR	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	D0 ← [MAR]	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Store	MAR ← IR	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	[MAR] ← D0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
Add	MAR ← IR	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	MBR ← [MAR]	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	ALU (P) ← MBR	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	D0 ← ALU	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
Sub	MAR ← IR	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	MBR ← [MAR]	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	ALU (P) ← MBR	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1
	D0 ← ALU	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
Inc	MAR ← IR	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	MBR ← [MAR]	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	ALU (P) ← MBR	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0
	[MAR] ← ALU	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
Dec	MAR ← IR	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	MBR ← [MAR]	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	ALU (P) ← MBR	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1
	[MAR] ← ALU	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
Bra	PC ← IR	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
Beq	If Z = 1 then PC ← IR	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0

Figure 2: Truth Table

The purpose of this learning activity is to create a control signal logic encoder where it takes in 15 inputs and has 15 outputs. The 15 inputs form different combinations for each of the microinstructions of the outputs. Figure 1 illustrates the schematic and Figure 2 shows the truth table for the learning activity.

<u>Control Signal Logic Signature</u>

- Input
  - Instructions: *load*, *store*, *add*, *sub*, *inc*, *dec*, *bra*, *beq*
  - Instruction Sequence Timing Signals: T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7
- Output
  - *R*, *w*, *cmar*, *cmbr*, *cpc*, *cir*, *cd0*, *calu*, *embr*, *epc*, *eir*, *ed0*, *ealu*, *f1*, *f0*

Verification Tests

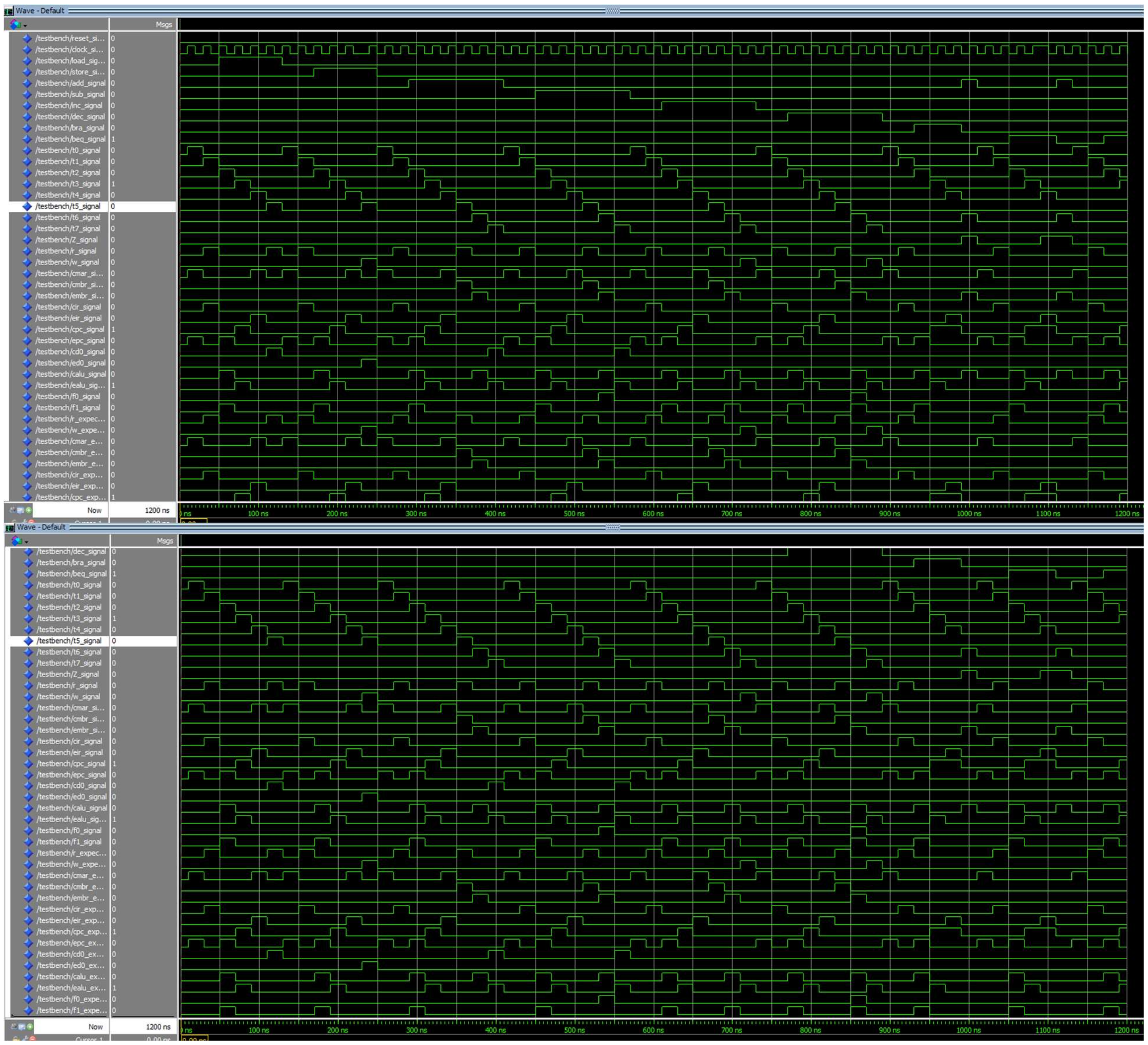
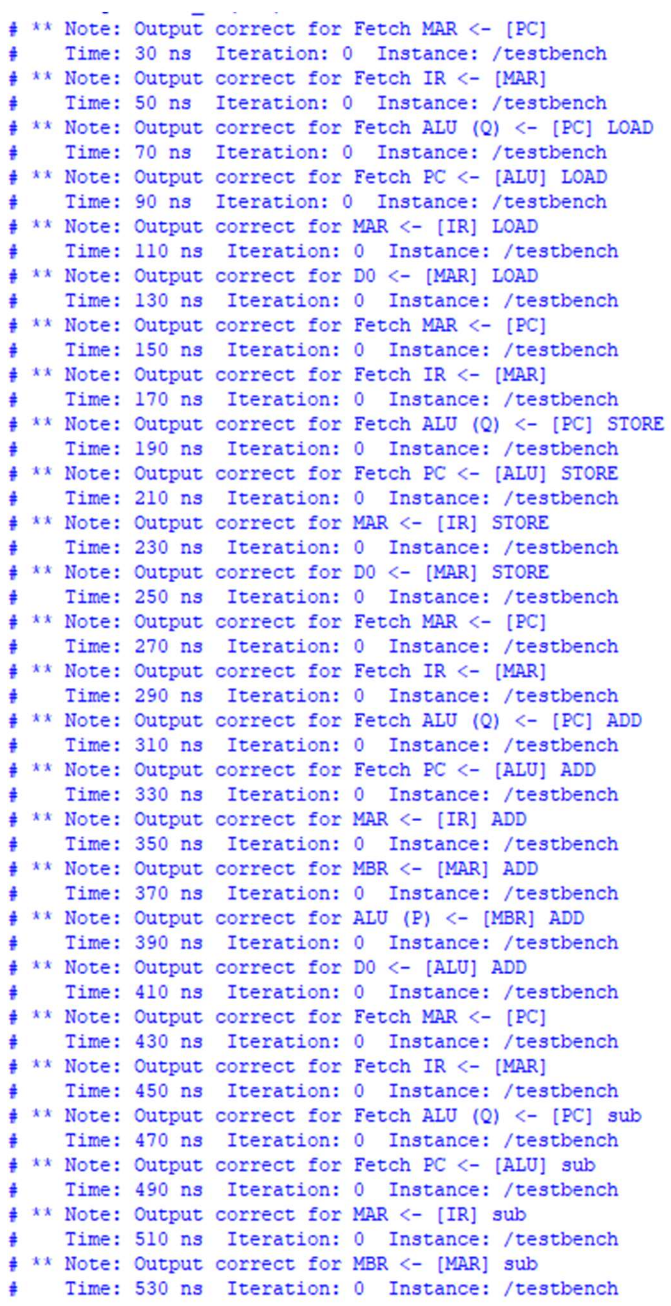
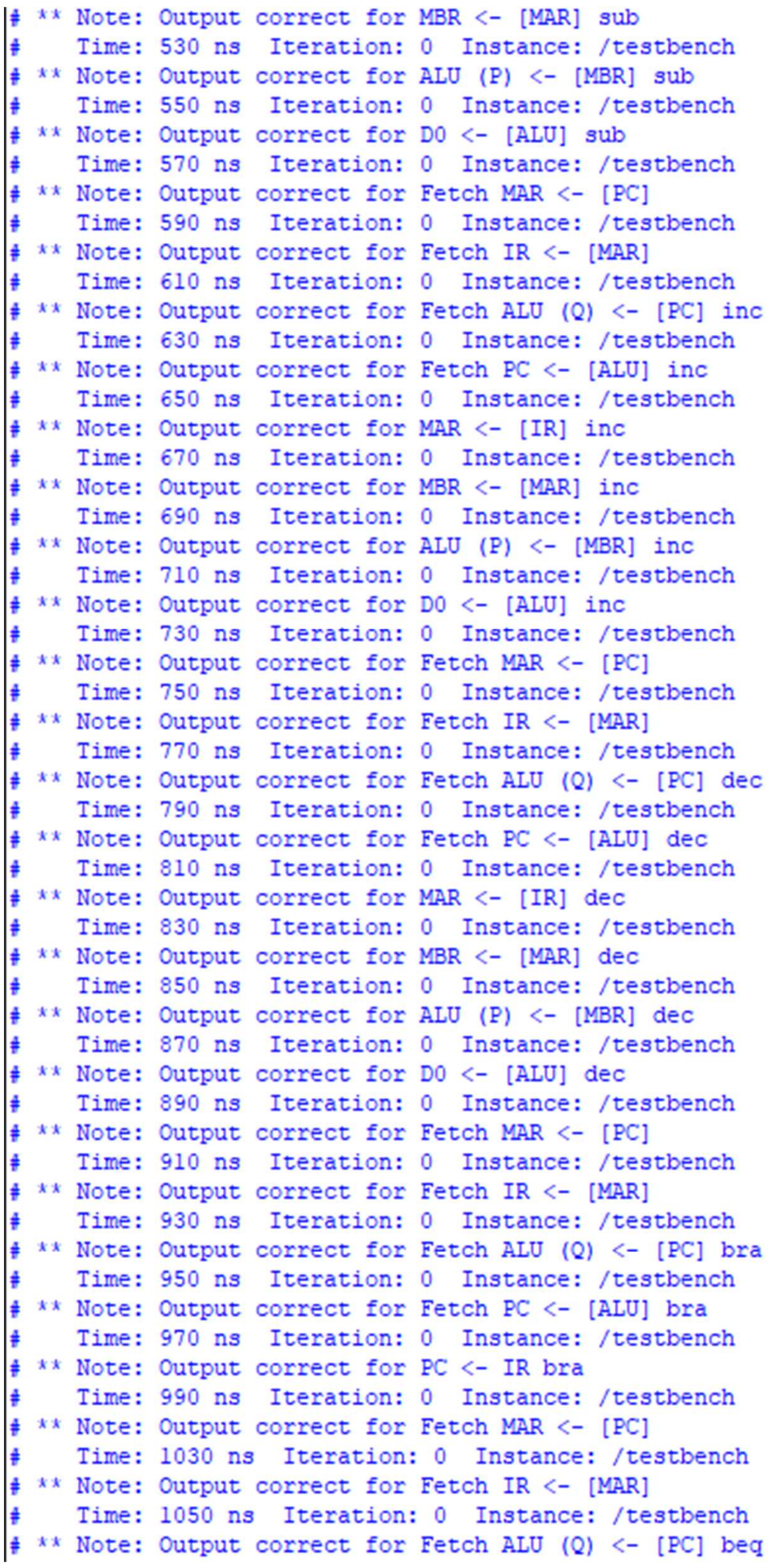


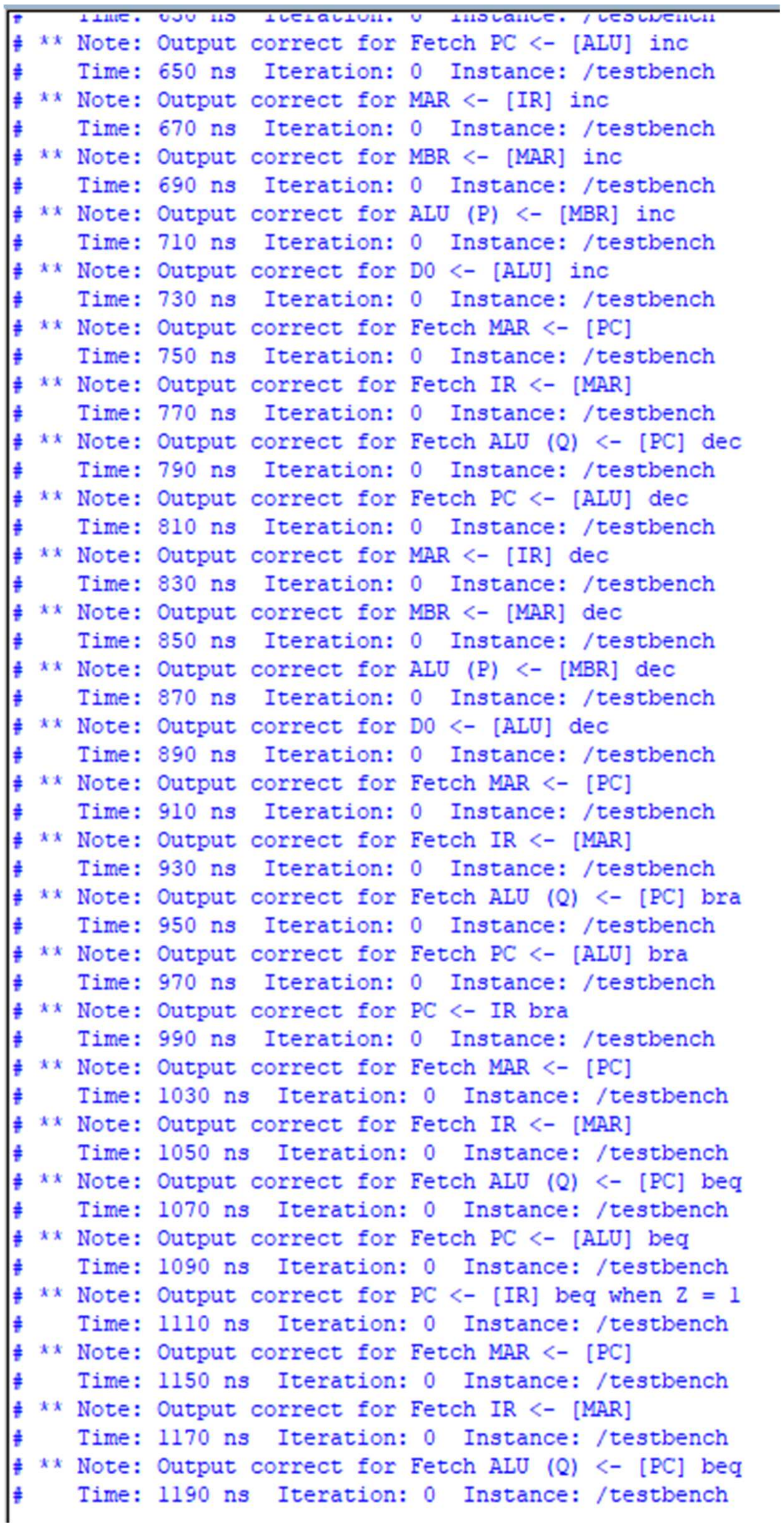
Figure 3: Verification Tests



```
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 30 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 50 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] LOAD
#    Time: 70 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] LOAD
#    Time: 90 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MAR <- [IR] LOAD
#    Time: 110 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for D0 <- [MAR] LOAD
#    Time: 130 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 150 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 170 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] STORE
#    Time: 190 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] STORE
#    Time: 210 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MAR <- [IR] STORE
#    Time: 230 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for D0 <- [MAR] STORE
#    Time: 250 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 270 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 290 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] ADD
#    Time: 310 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] ADD
#    Time: 330 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MAR <- [IR] ADD
#    Time: 350 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MBR <- [MAR] ADD
#    Time: 370 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for ALU (P) <- [MBR] ADD
#    Time: 390 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for D0 <- [ALU] ADD
#    Time: 410 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 430 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 450 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] sub
#    Time: 470 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] sub
#    Time: 490 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MAR <- [IR] sub
#    Time: 510 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MBR <- [MAR] sub
#    Time: 530 ns  Iteration: 0  Instance: /testbench
```



```
# ** Note: Output correct for MBR <- [MAR] sub
#    Time: 530 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for ALU (P) <- [MBR] sub
#    Time: 550 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for D0 <- [ALU] sub
#    Time: 570 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 590 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 610 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] inc
#    Time: 630 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] inc
#    Time: 650 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MAR <- [IR] inc
#    Time: 670 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MBR <- [MAR] inc
#    Time: 690 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for ALU (P) <- [MBR] inc
#    Time: 710 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for D0 <- [ALU] inc
#    Time: 730 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 750 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 770 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] dec
#    Time: 790 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] dec
#    Time: 810 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MAR <- [IR] dec
#    Time: 830 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MBR <- [MAR] dec
#    Time: 850 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for ALU (P) <- [MBR] dec
#    Time: 870 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for D0 <- [ALU] dec
#    Time: 890 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 910 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 930 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] bra
#    Time: 950 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] bra
#    Time: 970 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for PC <- IR bra
#    Time: 990 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 1030 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 1050 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] beq
```



```
#    Time: 630 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] inc
#    Time: 650 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MAR <- [IR] inc
#    Time: 670 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MBR <- [MAR] inc
#    Time: 690 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for ALU (P) <- [MBR] inc
#    Time: 710 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for D0 <- [ALU] inc
#    Time: 730 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 750 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 770 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] dec
#    Time: 790 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] dec
#    Time: 810 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MAR <- [IR] dec
#    Time: 830 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for MBR <- [MAR] dec
#    Time: 850 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for ALU (P) <- [MBR] dec
#    Time: 870 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for D0 <- [ALU] dec
#    Time: 890 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 910 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 930 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] bra
#    Time: 950 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] bra
#    Time: 970 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for PC <- IR bra
#    Time: 990 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 1030 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 1050 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] beq
#    Time: 1070 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch PC <- [ALU] beq
#    Time: 1090 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for PC <- [IR] beq when Z = 1
#    Time: 1110 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch MAR <- [PC]
#    Time: 1150 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch IR <- [MAR]
#    Time: 1170 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for Fetch ALU (Q) <- [PC] beq
#    Time: 1190 ns  Iteration: 0  Instance: /testbench
```

Figure 4: Verification Tests with Assert Statements

Figures 3-4 display the verification and testing process of the control signal logic schematic. The test results indicate that the schematic passed all tests according to the verification tests and assert statements. Therefore, the control signal logic schematic is correct and successfully passed all the tests.