Tyler Kim

Digital Logic Design

Opcode Decoder Verification Report

Component

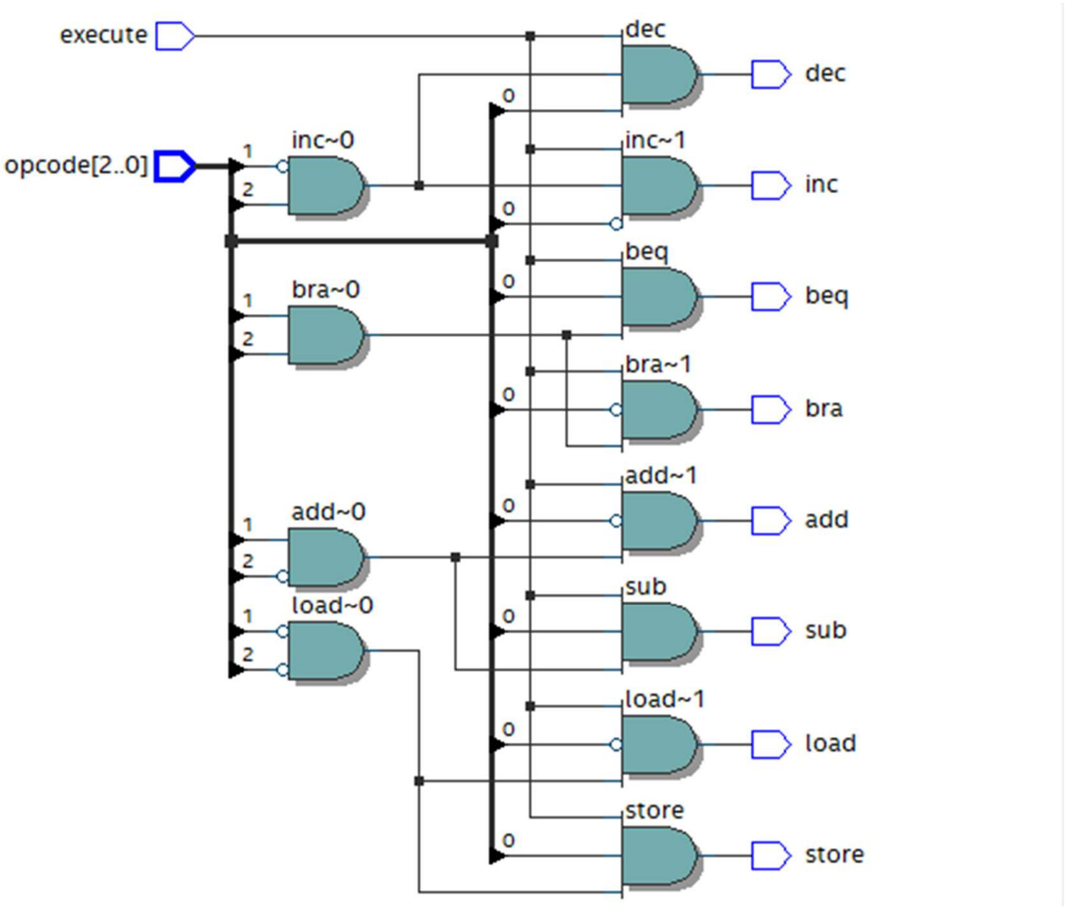


Figure 1: Opcode-Decoder Visual Schematic

Opcode	**Output**
000	Load
001	Store
010	Add
011	Sub
100	Inc
101	Dec
110	Bra
111	Beq

Figure 2: Mapping of Opcode to Output

The purpose of this learning activity is to create a opcode-decoder combinational circuit where it takes in an input called *execute* and a bus with 3 digits called *opcode*. Whenever *execute* is 0, all outputs will print out 0. If *execute* is 1 then the affiliated *opcode* value will choose which output to send a signal of 1. Figure 1 illustrates the visual schematic of the opcode-decoder. Figure 2 shows the mapping between the opcode and the respective output.

Opcode-Decoder Signature

- INPUTS
    - Execute
    - Opcode from 0 - 2
- OUTPUTS
    - Load
    - Store

- Add
- Sub
- Inc
- Dec
- Bra
- beq

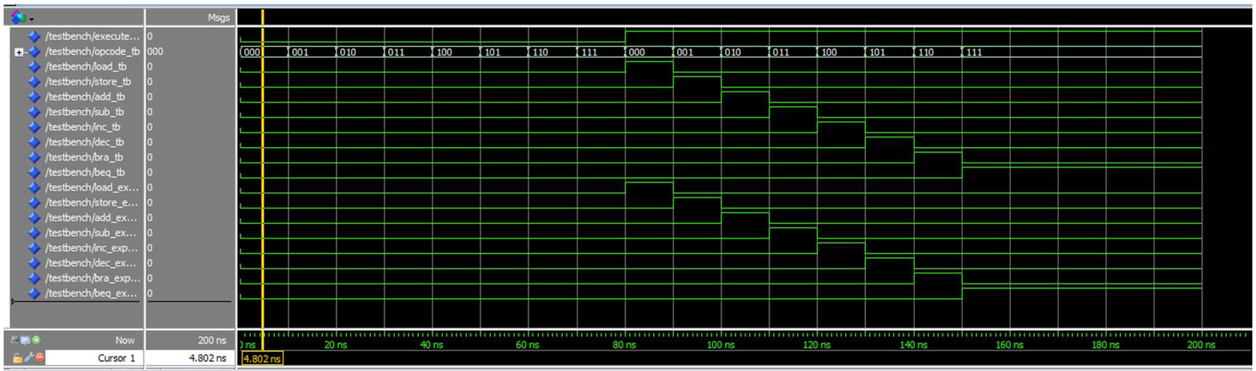
Verification Tests

Port Mapping
execute = execute_tb
opcode[2..0] = opcode_tb
load = load_tb
store = store_tb
add = add_tb
sub = sub_tb
inc = inc_tb
dec = dec_tb
bra = bra_tb
beq = beq_tb

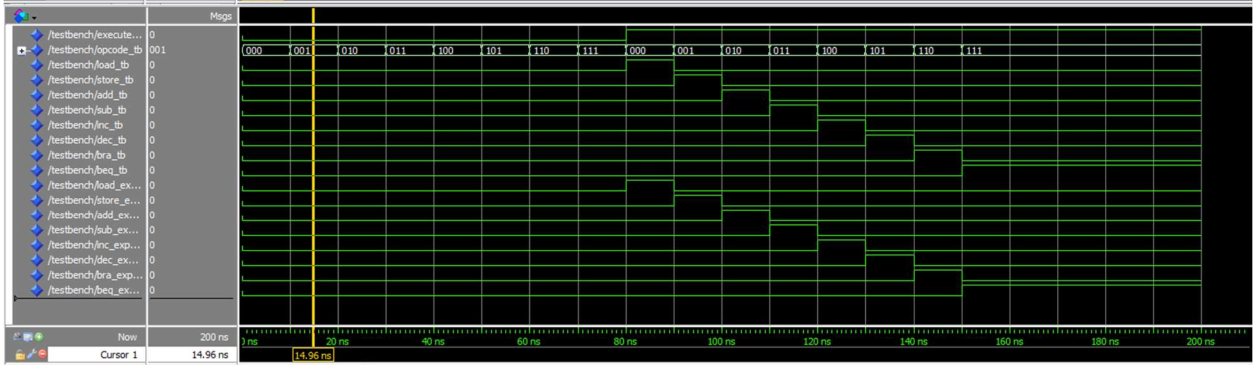
execute_tb	**opcode_tb**	**load**	**store**	**add**	**sub**	**inc**	**dec**	**bra**	**beq**
0	000	0	0	0	0	0	0	0	0
0	001	0	0	0	0	0	0	0	0
0	010	0	0	0	0	0	0	0	0
0	011	0	0	0	0	0	0	0	0
0	100	0	0	0	0	0	0	0	0
0	101	0	0	0	0	0	0	0	0
0	110	0	0	0	0	0	0	0	0

0	111	0	0	0	0	0	0	0	0
1	000	1	0	0	0	0	0	0	0
1	001	0	1	0	0	0	0	0	0
1	010	0	0	1	0	0	0	0	0
1	011	0	0	0	1	0	0	0	0
1	100	0	0	0	0	1	0	0	0
1	101	0	0	0	0	0	1	0	0
1	110	0	0	0	0	0	0	1	0
1	111	0	0	0	0	0	0	0	1

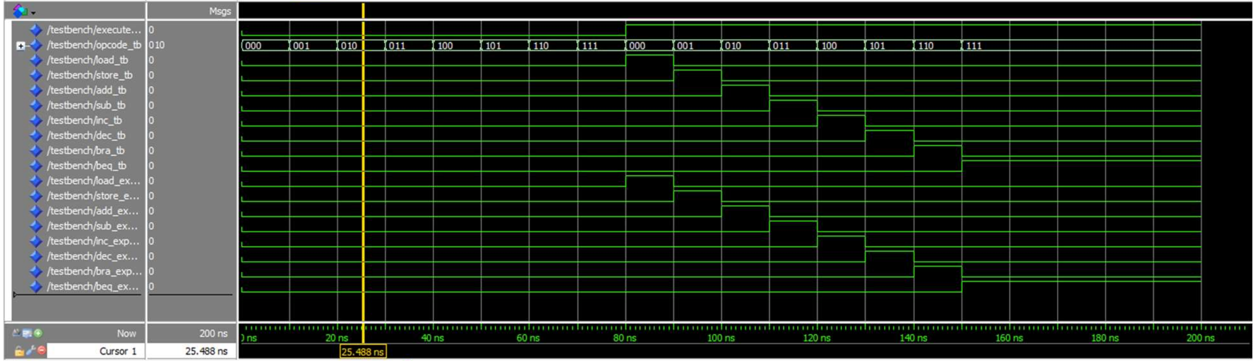
Figure 3: Truth Table of the Opcode-Decoder



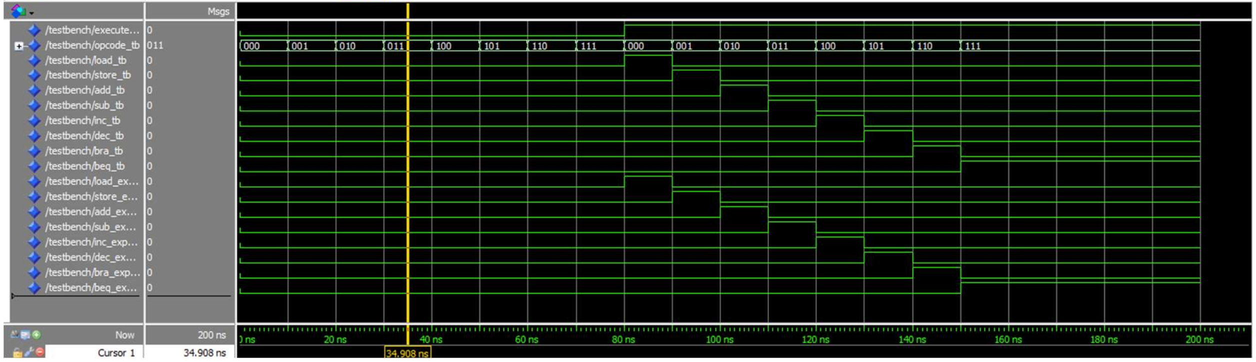
	Msgs
/testbench/execute...	0
/testbench/opcode_tb	000
/testbench/load_tb	0
/testbench/store_tb	0
/testbench/add_tb	0
/testbench/sub_tb	0
/testbench/inc_tb	0
/testbench/dec_tb	0
/testbench/bra_tb	0
/testbench/beq_tb	0
/testbench/load_ex...	0
/testbench/store_e...	0
/testbench/add_ex...	0
/testbench/sub_ex...	0
/testbench/inc_exp...	0
/testbench/dec_ex...	0
/testbench/bra_exp...	0
/testbench/beq_ex...	0
Now	200 ns
Cursor 1	4.802 ns



	Msgs
/testbench/execute...	0
/testbench/opcode_tb	001
/testbench/load_tb	0
/testbench/store_tb	0
/testbench/add_tb	0
/testbench/sub_tb	0
/testbench/inc_tb	0
/testbench/dec_tb	0
/testbench/bra_tb	0
/testbench/beq_tb	0
/testbench/load_ex...	0
/testbench/store_e...	0
/testbench/add_ex...	0
/testbench/sub_ex...	0
/testbench/inc_exp...	0
/testbench/dec_ex...	0
/testbench/bra_exp...	0
/testbench/beq_ex...	0
Now	200 ns
Cursor 1	14.96 ns



	Msgs
/testbench/execute...	0
/testbench/opcode_tb	010
/testbench/load_tb	0
/testbench/store_tb	0
/testbench/add_tb	0
/testbench/sub_tb	0
/testbench/inc_tb	0
/testbench/dec_tb	0
/testbench/bra_tb	0
/testbench/beq_tb	0
/testbench/load_ex...	0
/testbench/store_e...	0
/testbench/add_ex...	0
/testbench/sub_ex...	0
/testbench/inc_exp...	0
/testbench/dec_ex...	0
/testbench/bra_exp...	0
/testbench/beq_ex...	0
Now	200 ns
Cursor 1	25.488 ns



	Msgs
/testbench/execute...	0
/testbench/opcode_tb	011
/testbench/load_tb	0
/testbench/store_tb	0
/testbench/add_tb	0
/testbench/sub_tb	0
/testbench/inc_tb	0
/testbench/dec_tb	0
/testbench/bra_tb	0
/testbench/beq_tb	0
/testbench/load_ex...	0
/testbench/store_e...	0
/testbench/add_ex...	0
/testbench/sub_ex...	0
/testbench/inc_exp...	0
/testbench/dec_ex...	0
/testbench/bra_exp...	0
/testbench/beq_ex...	0
Now	200 ns
Cursor 1	34.908 ns

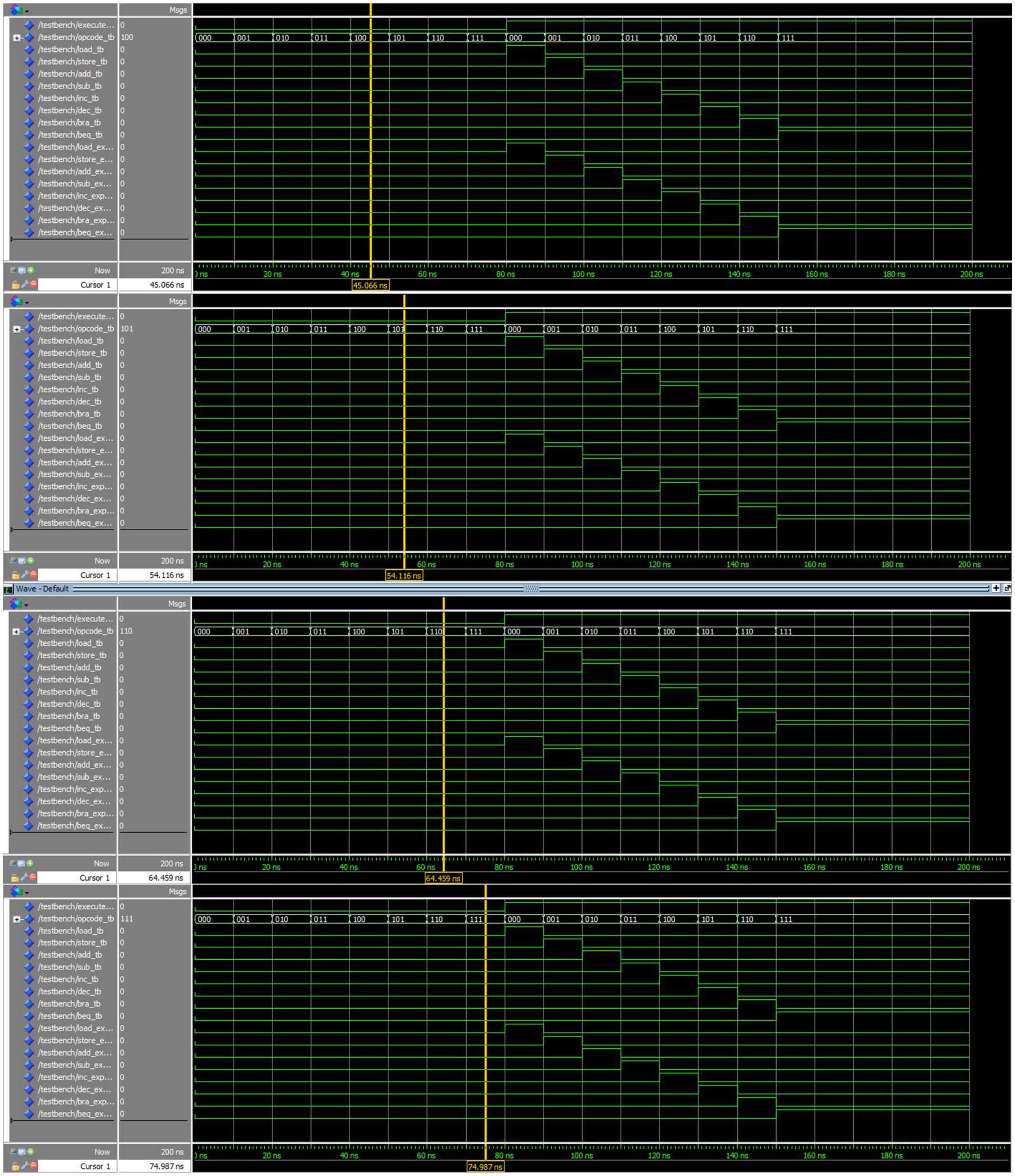
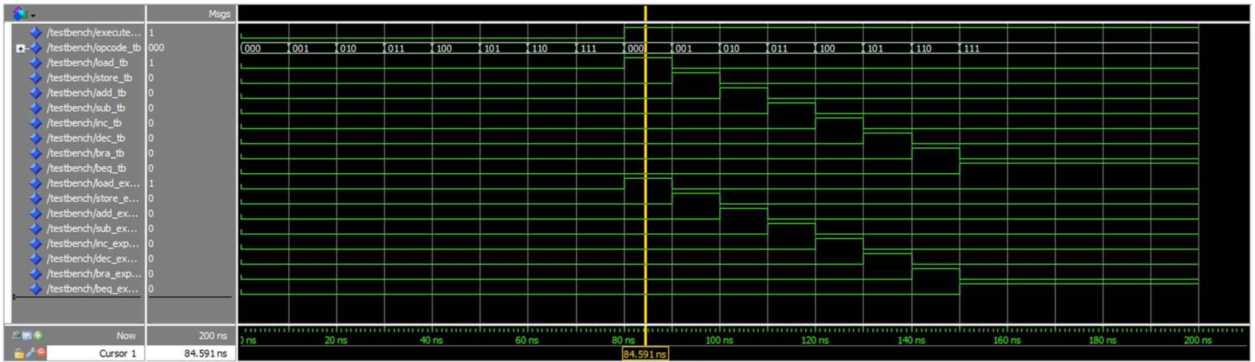


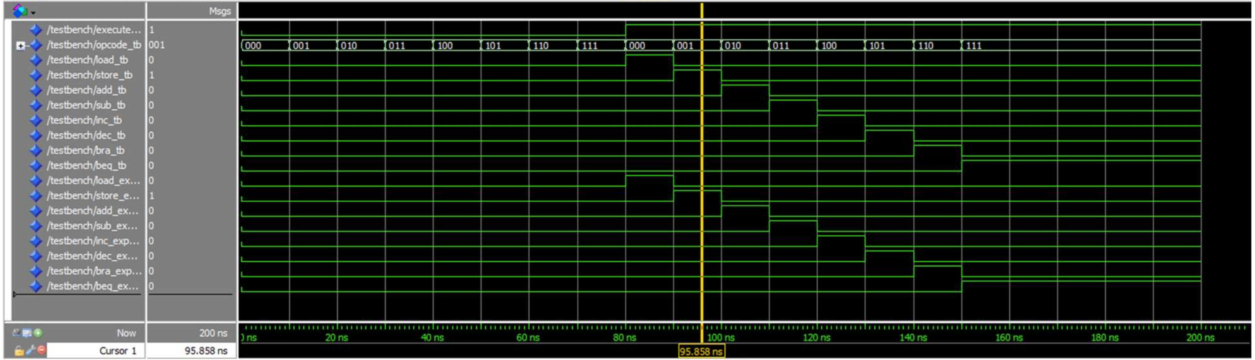
Figure 4: Verification Tests when *execute* = 0



Msgs

/testbench/execute… 1
/testbench/opcode_tb 000
/testbench/load_tb 1
/testbench/store_tb 0
/testbench/add_tb 0
/testbench/sub_tb 0
/testbench/inc_tb 0
/testbench/dec_tb 0
/testbench/bra_tb 0
/testbench/beq_tb 0
/testbench/load_ex… 1
/testbench/store_e… 0
/testbench/add_ex… 0
/testbench/sub_ex… 0
/testbench/inc_exp… 0
/testbench/dec_ex… 0
/testbench/bra_exp… 0
/testbench/beq_ex… 0

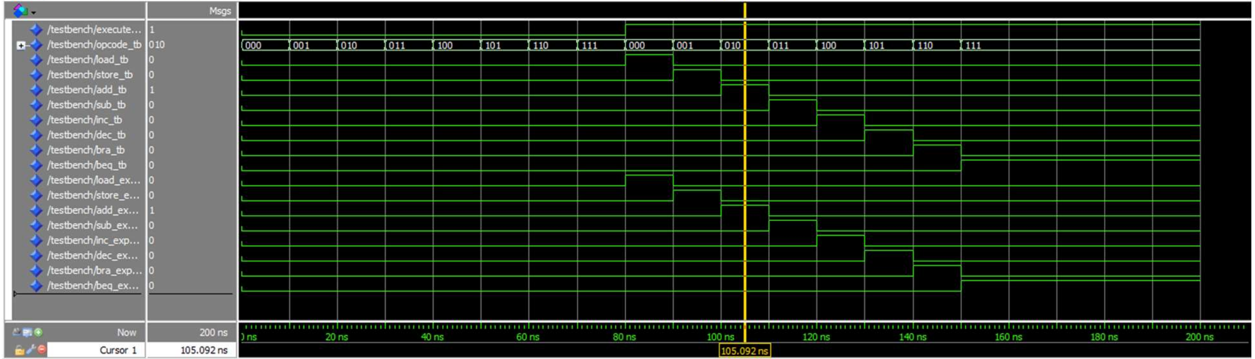
Now 200 ns
Cursor 1 84.591 ns



Msgs

/testbench/execute… 1
/testbench/opcode_tb 001
/testbench/load_tb 0
/testbench/store_tb 1
/testbench/add_tb 0
/testbench/sub_tb 0
/testbench/inc_tb 0
/testbench/dec_tb 0
/testbench/bra_tb 0
/testbench/beq_tb 0
/testbench/load_ex… 0
/testbench/store_e… 1
/testbench/add_ex… 0
/testbench/sub_ex… 0
/testbench/inc_exp… 0
/testbench/dec_ex… 0
/testbench/bra_exp… 0
/testbench/beq_ex… 0

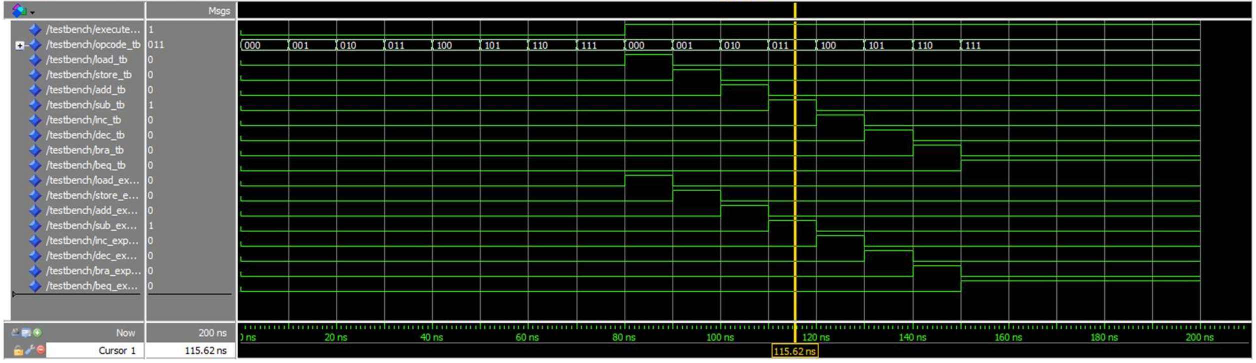
Now 200 ns
Cursor 1 95.858 ns



Msgs

/testbench/execute… 1
/testbench/opcode_tb 010
/testbench/load_tb 0
/testbench/store_tb 0
/testbench/add_tb 1
/testbench/sub_tb 0
/testbench/inc_tb 0
/testbench/dec_tb 0
/testbench/bra_tb 0
/testbench/beq_tb 0
/testbench/load_ex… 0
/testbench/store_e… 0
/testbench/add_ex… 1
/testbench/sub_ex… 0
/testbench/inc_exp… 0
/testbench/dec_ex… 0
/testbench/bra_exp… 0
/testbench/beq_ex… 0

Now 200 ns
Cursor 1 105.092 ns



Msgs

/testbench/execute… 1
/testbench/opcode_tb 011
/testbench/load_tb 0
/testbench/store_tb 0
/testbench/add_tb 0
/testbench/sub_tb 1
/testbench/inc_tb 0
/testbench/dec_tb 0
/testbench/bra_tb 0
/testbench/beq_tb 0
/testbench/load_ex… 0
/testbench/store_e… 0
/testbench/add_ex… 0
/testbench/sub_ex… 1
/testbench/inc_exp… 0
/testbench/dec_ex… 0
/testbench/bra_exp… 0
/testbench/beq_ex… 0

Now 200 ns
Cursor 1 115.62 ns

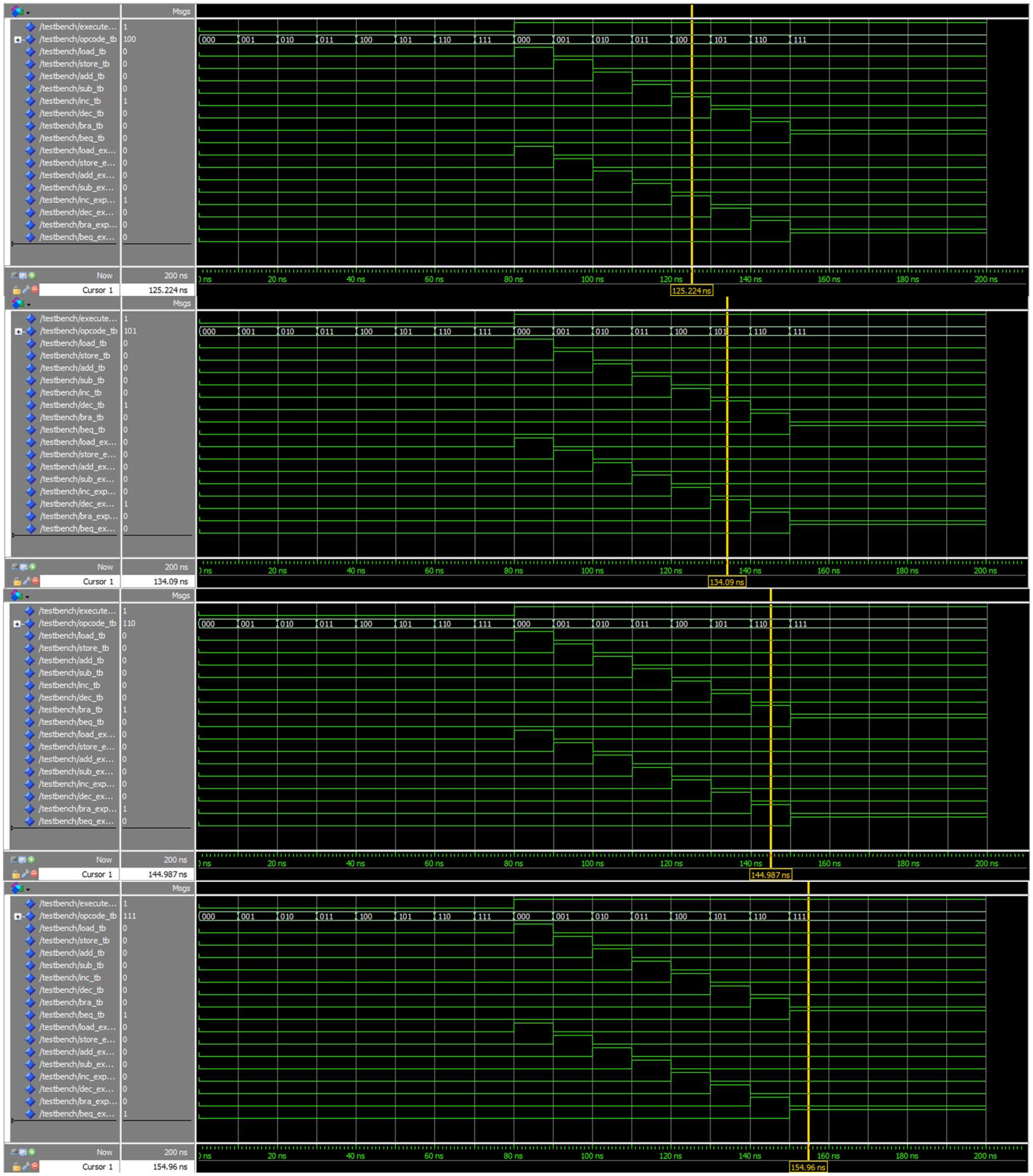
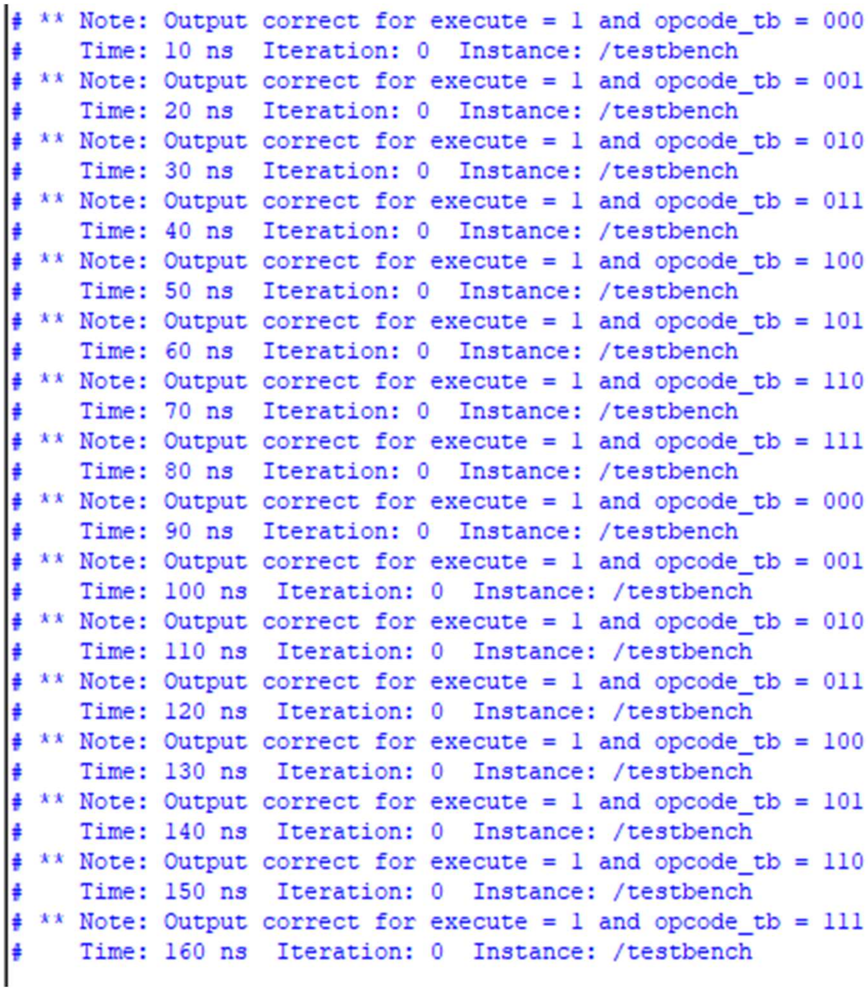


Figure 5: Verification Test when *execute* = 1



```
# ** Note: Output correct for execute = 1 and opcode_tb = 000
#    Time: 10 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 001
#    Time: 20 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 010
#    Time: 30 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 011
#    Time: 40 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 100
#    Time: 50 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 101
#    Time: 60 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 110
#    Time: 70 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 111
#    Time: 80 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 000
#    Time: 90 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 001
#    Time: 100 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 010
#    Time: 110 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 011
#    Time: 120 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 100
#    Time: 130 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 101
#    Time: 140 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 110
#    Time: 150 ns  Iteration: 0  Instance: /testbench
# ** Note: Output correct for execute = 1 and opcode_tb = 111
#    Time: 160 ns  Iteration: 0  Instance: /testbench
```

Figure 6: Verification Test Using Assert Statements

Figures 4 – 6 display the verification and testing process of the opcode-decoder schematic. The test results in indicate that the schematic successfully passed according to the truth table in Figure 3. Therefore, the opcode-decoder schematic is correct and passed all the tests.