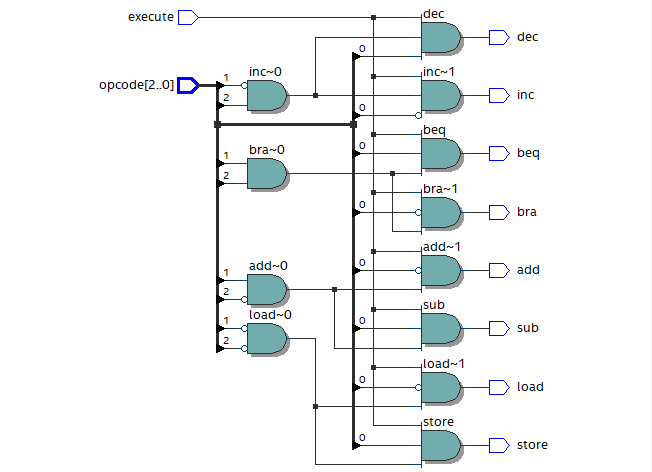
Tyler Kim

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

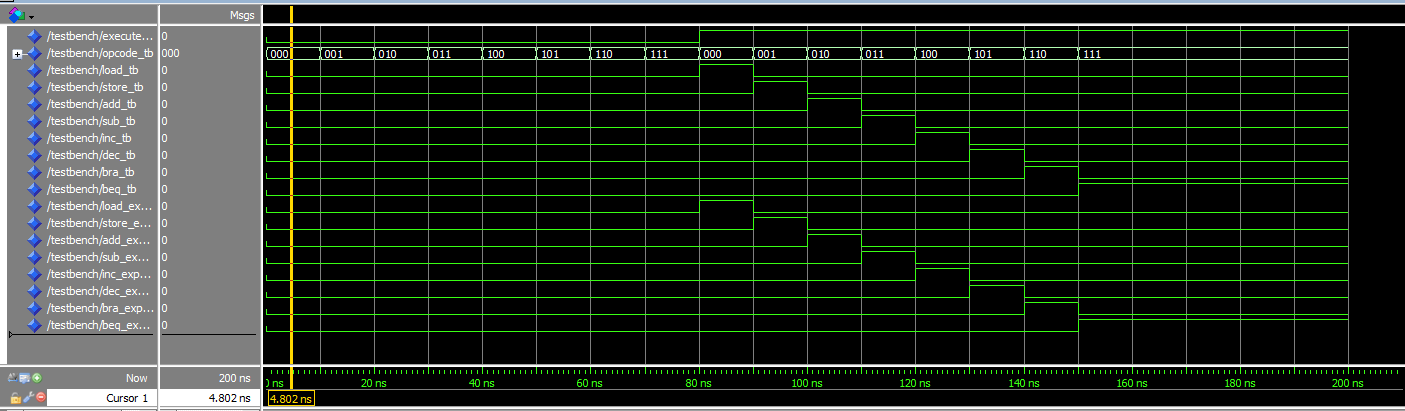
dec = dec\_tb

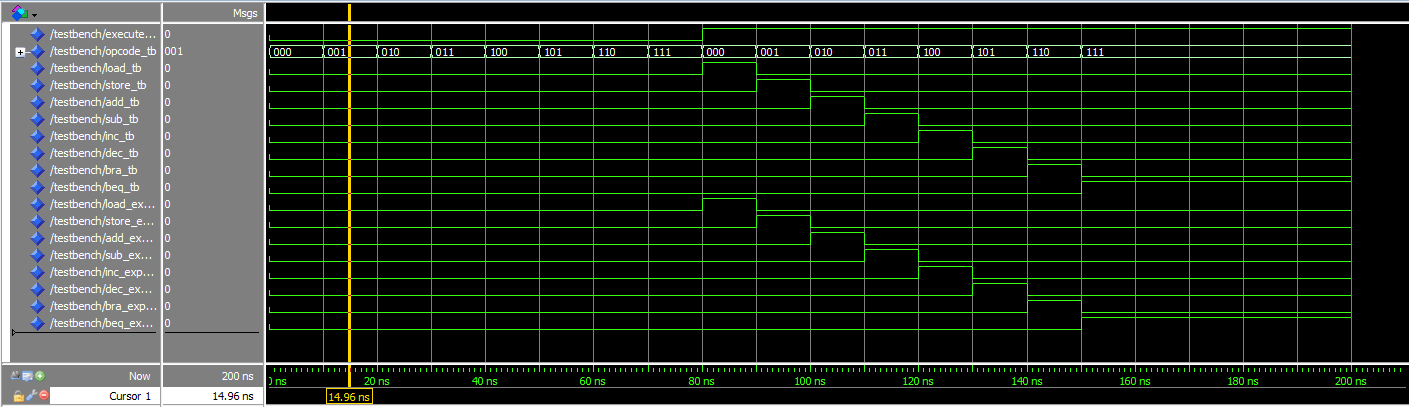
bra = bra\_tb

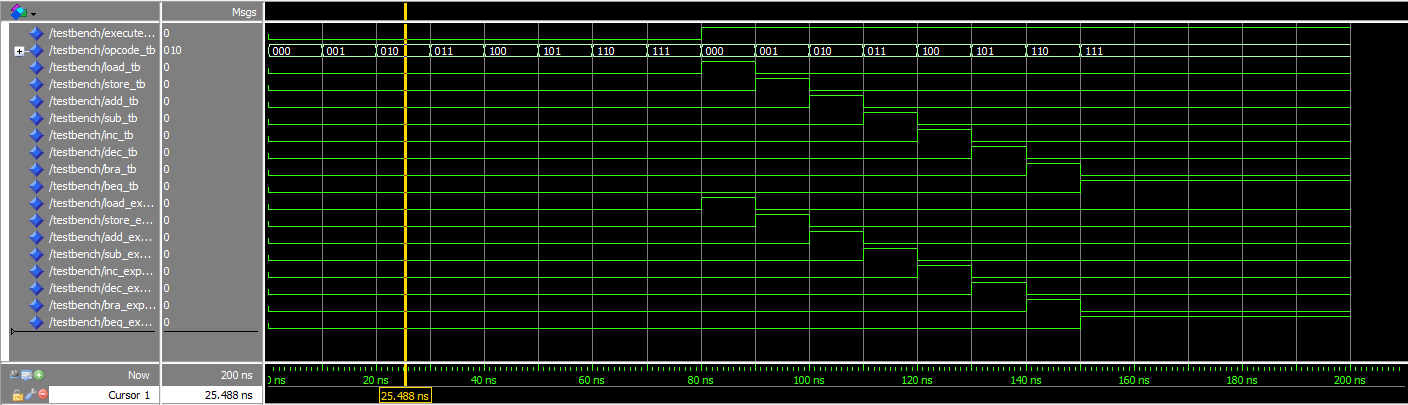
beq = beq\_tb

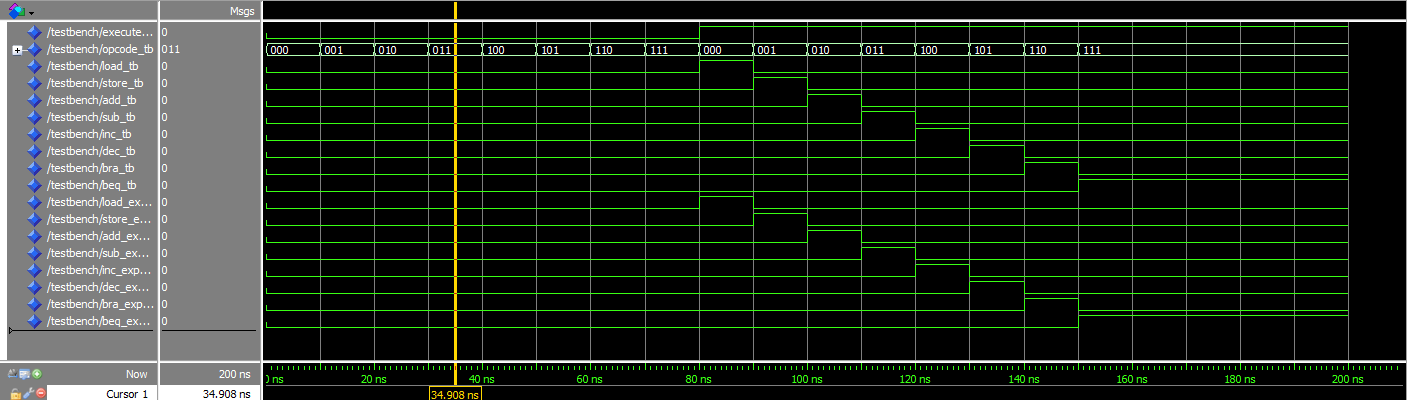
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **execute\_tb** | **opcode\_tb** | **load** | **store** | **add** | **sub** | **inc** | **dec** | **bra** | **beq** |
| 0 | 000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 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 |

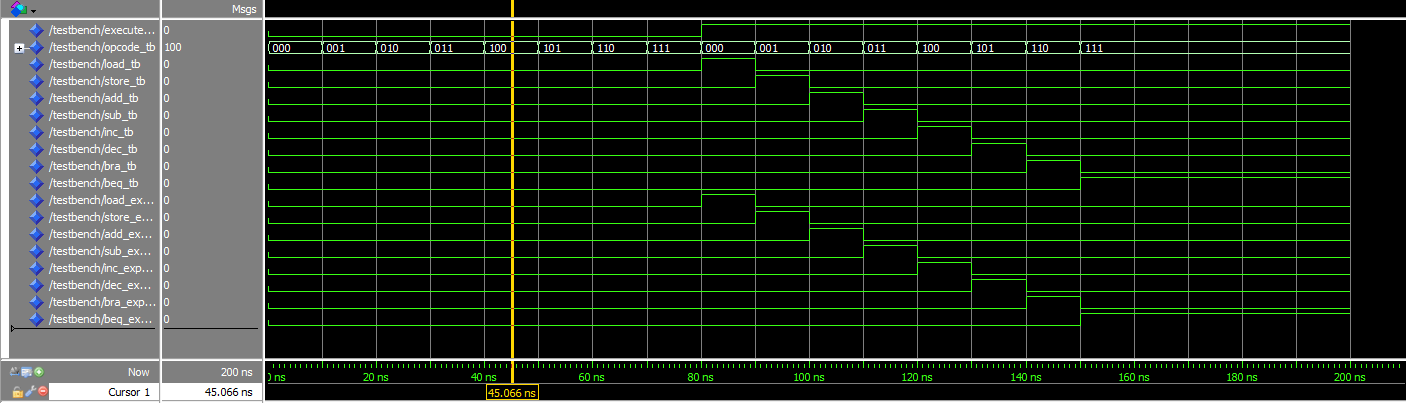
**Figure 3: Truth Table of the Opcode-Decoder**

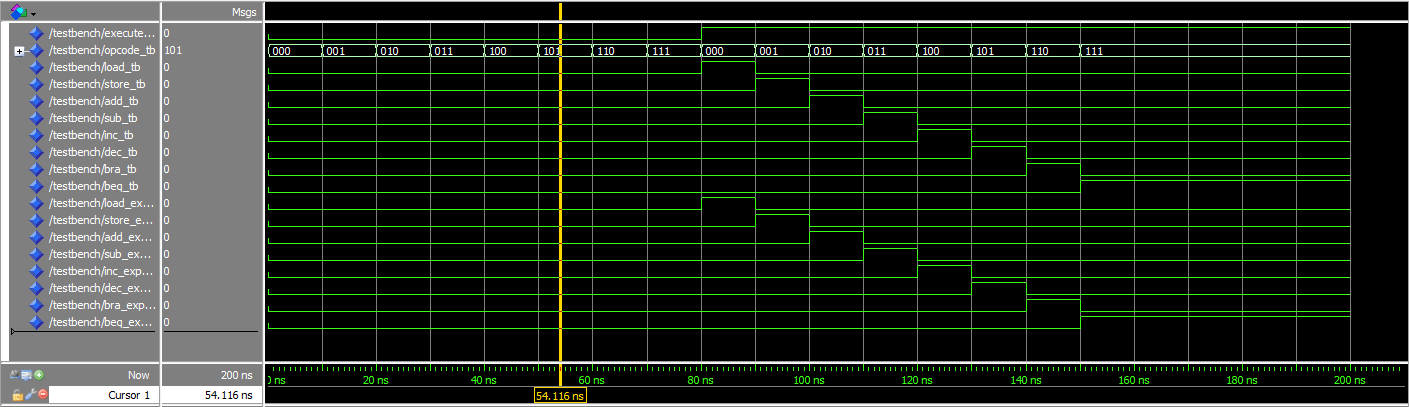
****

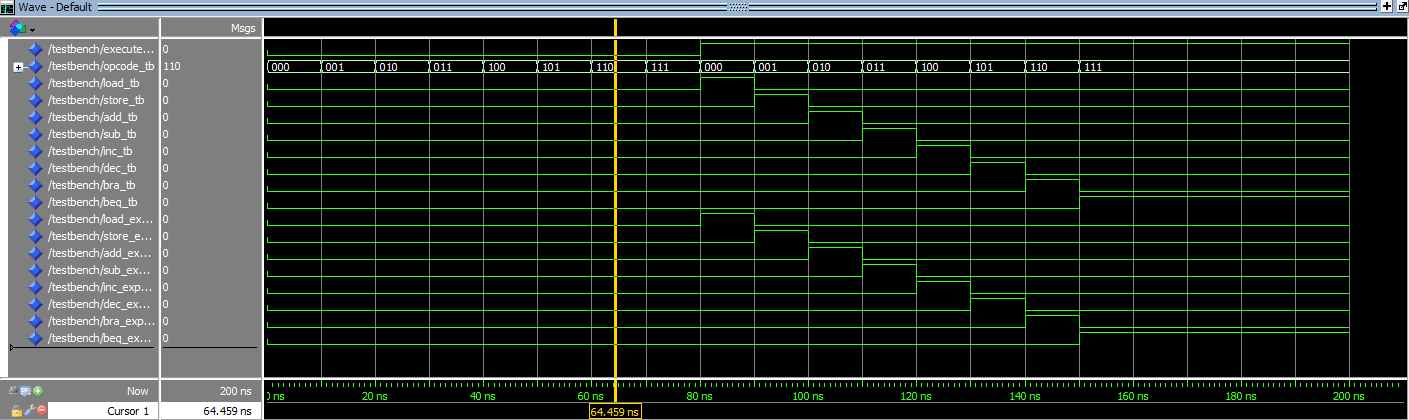
****

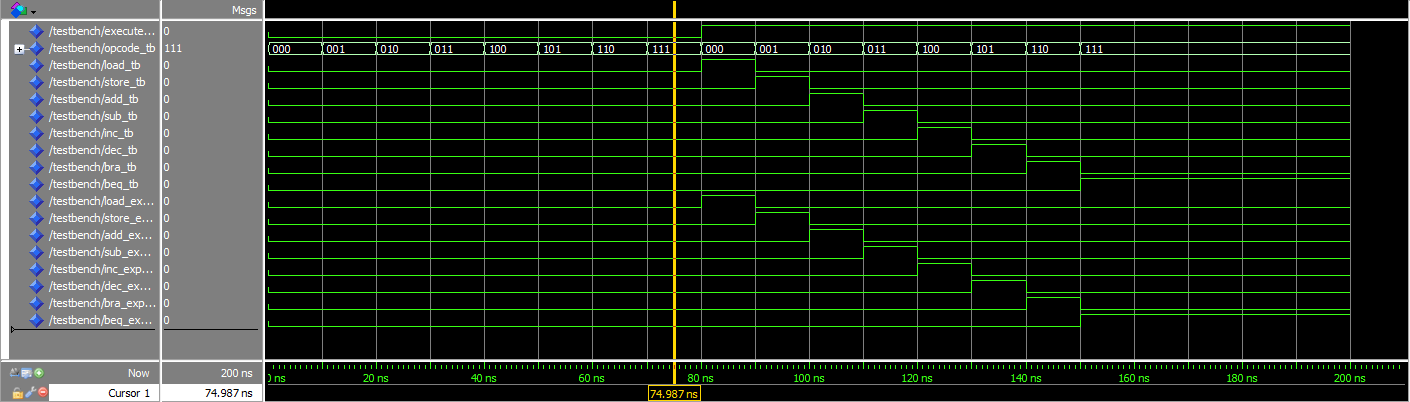
****

****

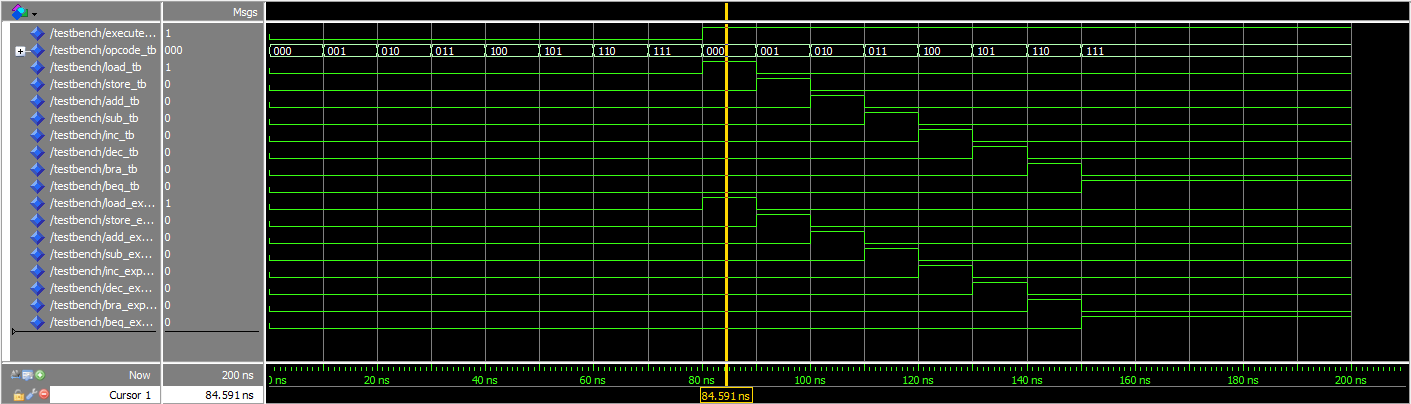
****

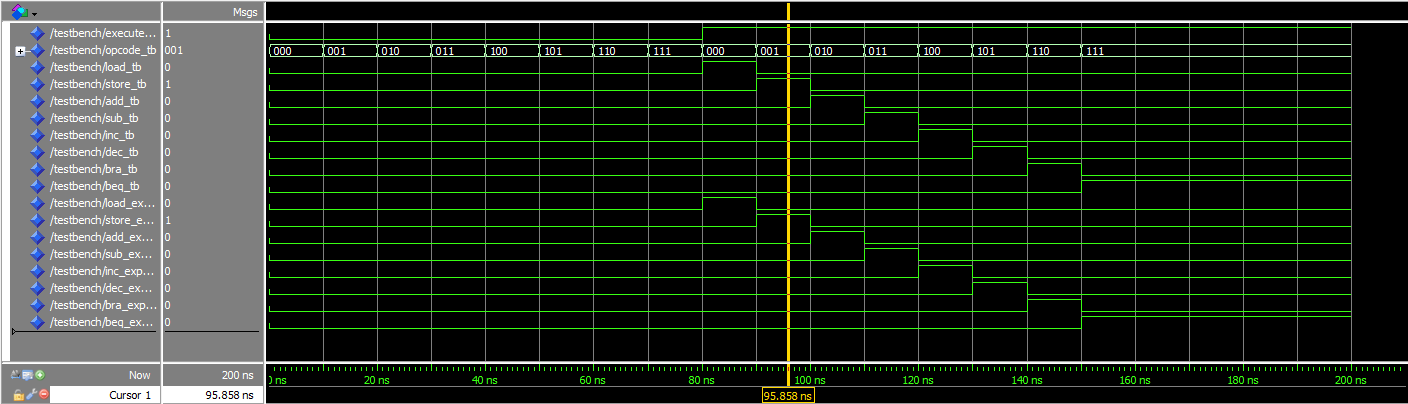
****

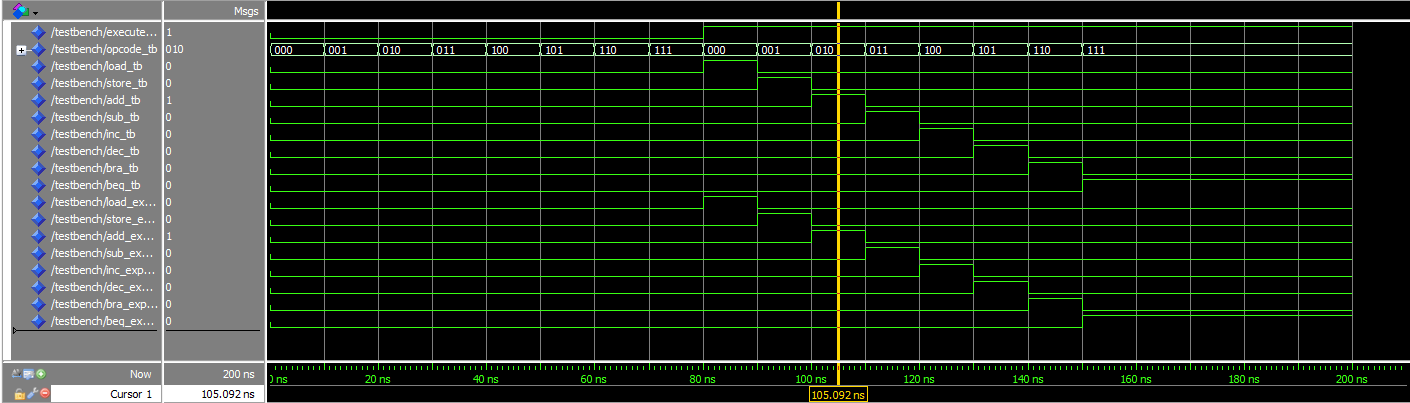
****

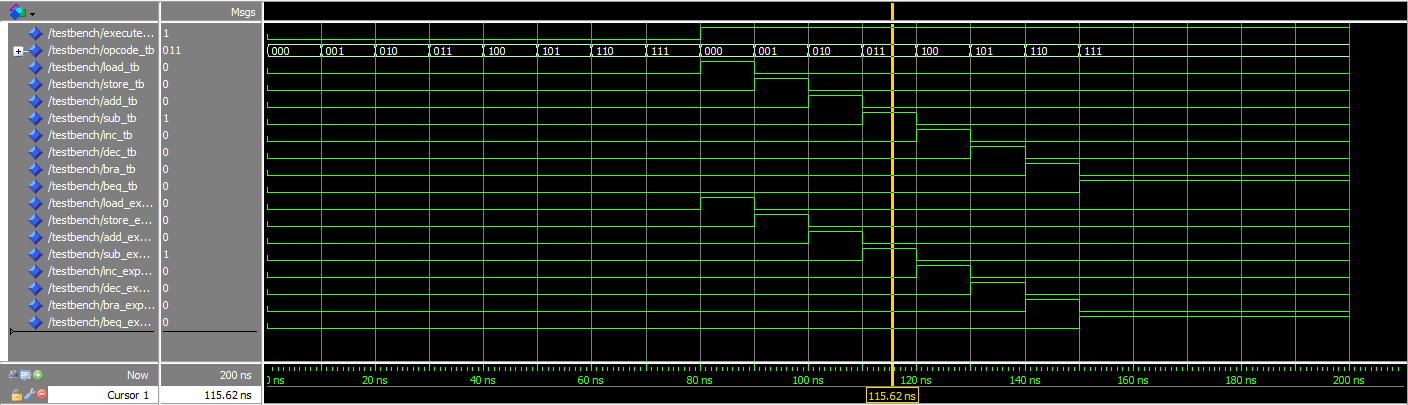
****

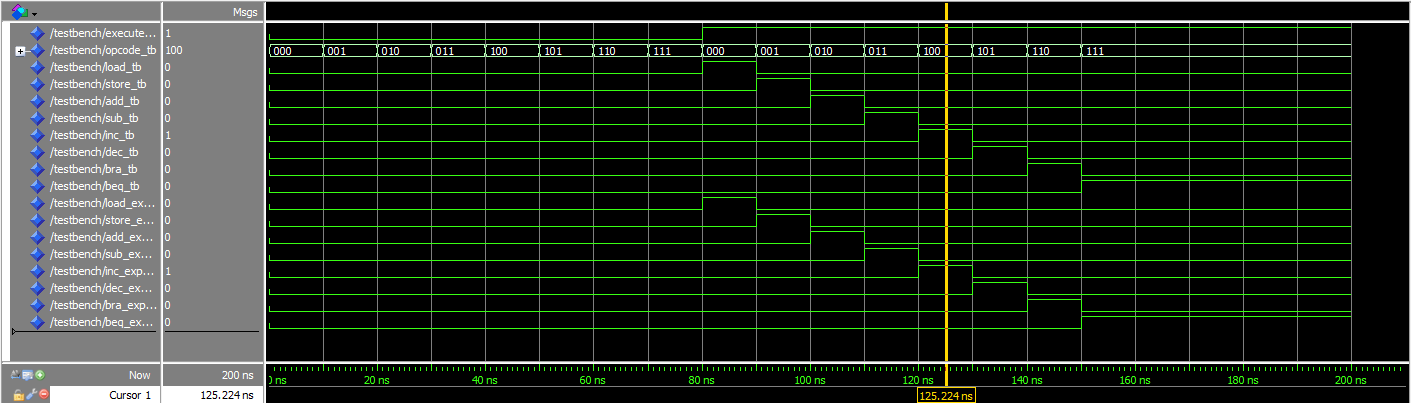
**Figure 4: Verification Tests when *execute* = 0**

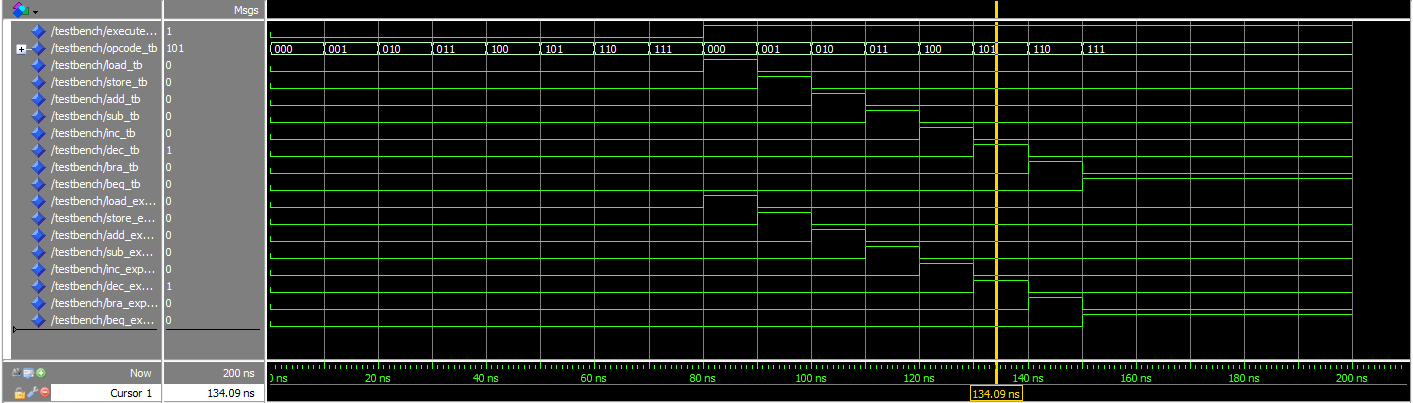
****

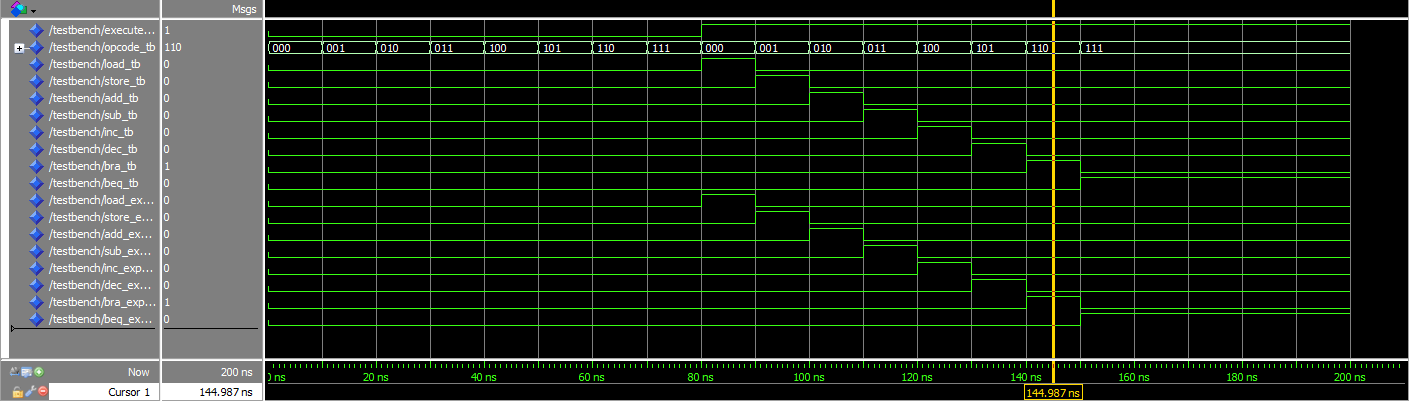
****

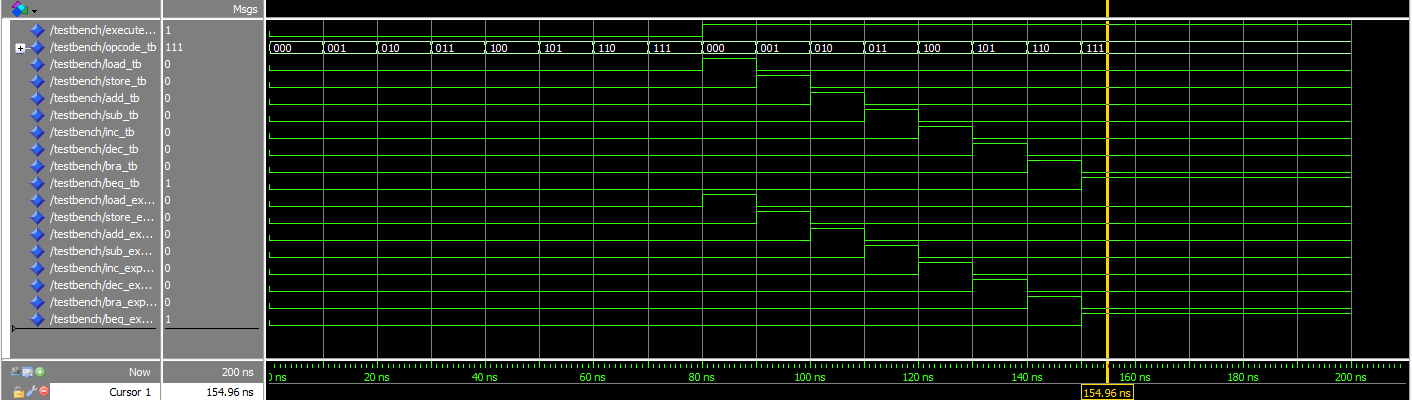
****

****

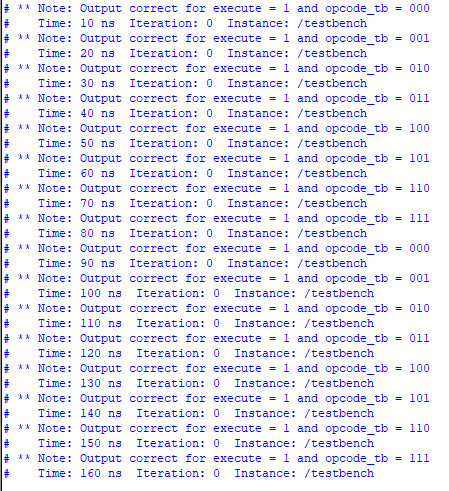
****

****

****

****

**Figure 5: Verification Test when *execute* = 1**

****

**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.