ECE 6276
DSP Hardware System Design
Fall 2017

Lab 1

William Sutton

September 7, 2017

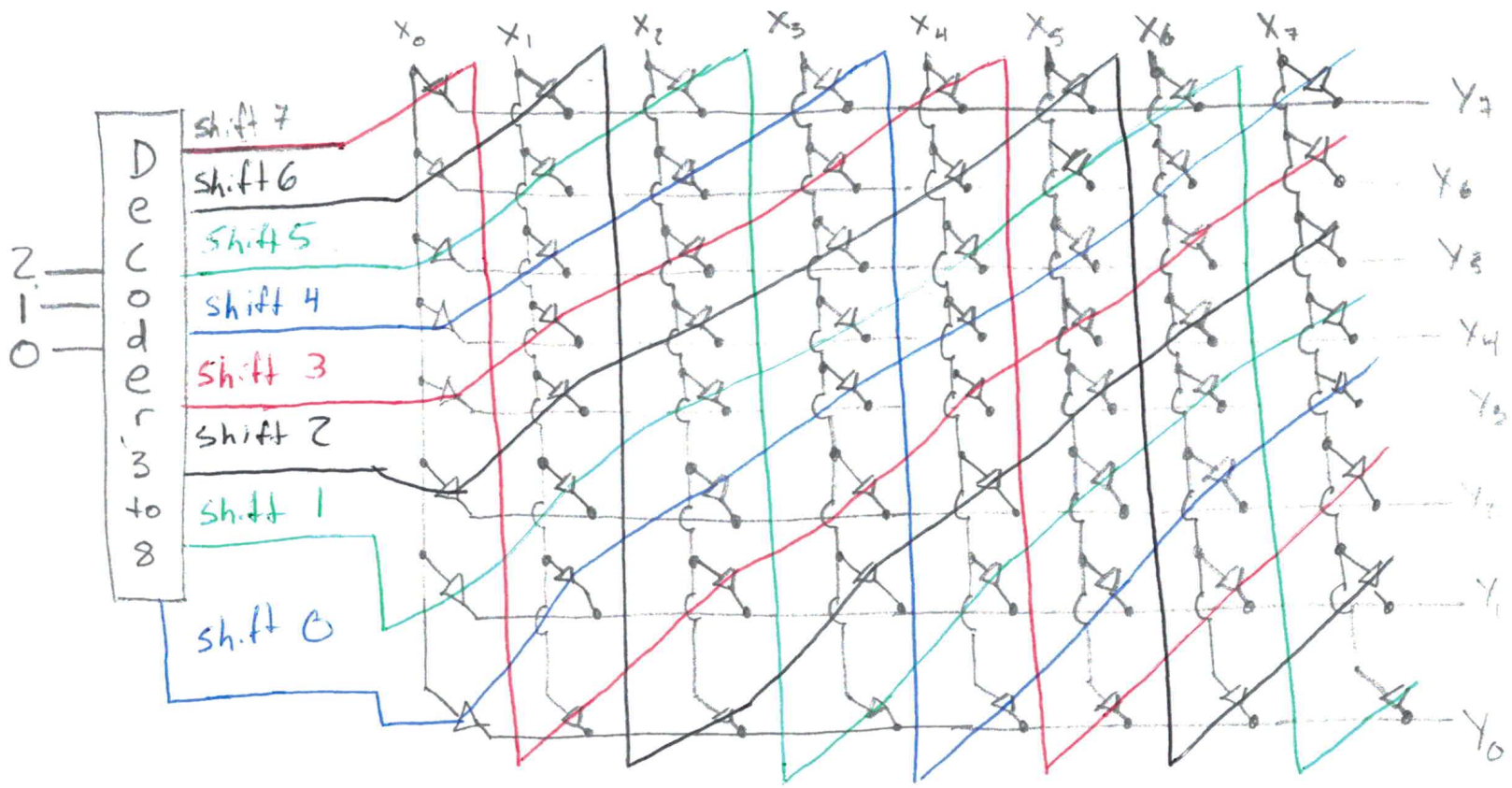
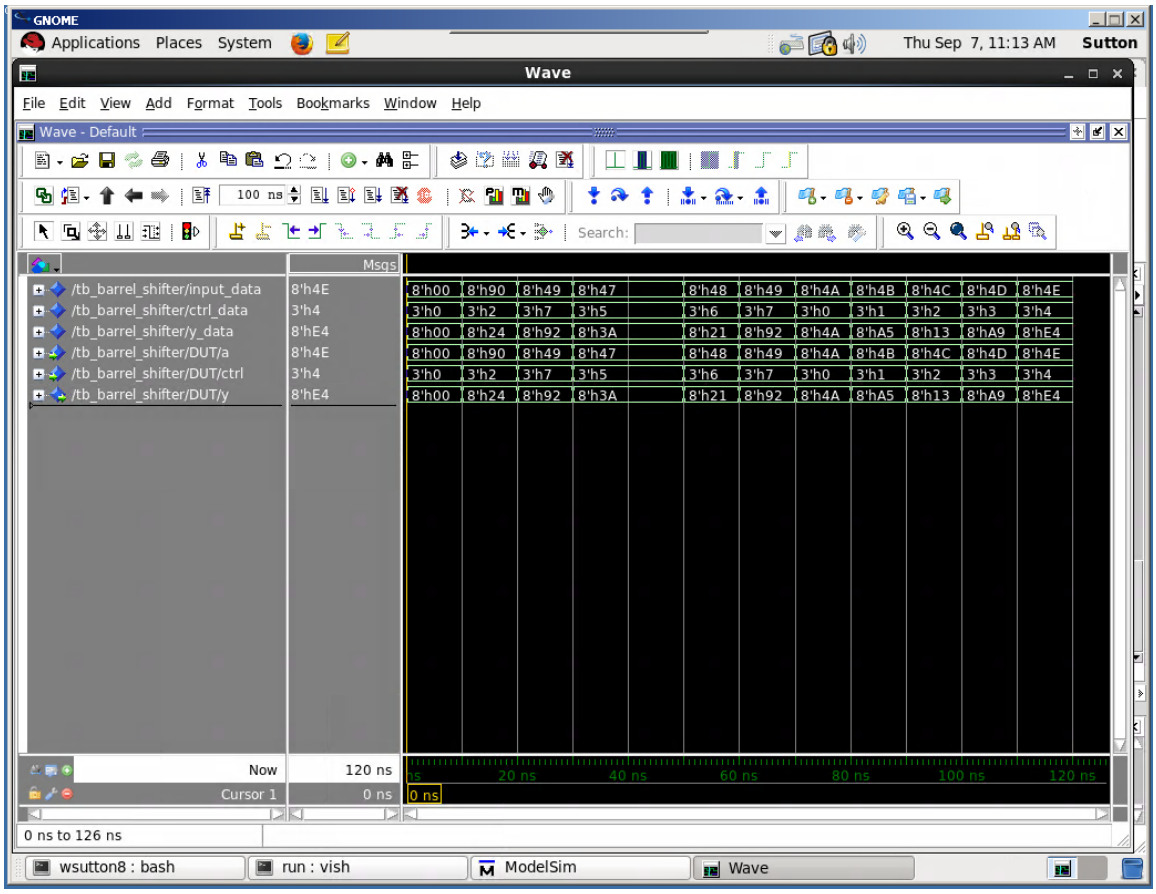

Decoder 3 to 8
Z
1
0
shift 7
shift 6
shift 5
shift 4
shift 3
shift 2
shift 1
shift 0
X0
X1
X2
X3
X4
X5
X6
X7
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0

Figure 1


GNOME
Applications Places System
Thu Sep 7, 11:13 AM Sutton
Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
100 ns
Search:
Msgs
/tb_barrel_shifter/input_data 8'h4E
/tb_barrel_shifter/ctrl_data 3'h4
/tb_barrel_shifter/y_data 8'hE4
/tb_barrel_shifter/DUT/a 8'h4E
/tb_barrel_shifter/DUT/ctrl 3'h4
/tb_barrel_shifter/DUT/y 8'hE4
8'h00 8'h90 8'h49 8'h47 8'h48 8'h49 8'h4A 8'h4B 8'h4C 8'h4D 8'h4E
3'h0 3'h2 3'h7 3'h5 3'h6 3'h7 3'h0 3'h1 3'h2 3'h3 3'h4
8'h00 8'h24 8'h92 8'h3A 8'h21 8'h92 8'h4A 8'hA5 8'h13 8'hA9 8'hE4
Now 120 ns
Cursor 1 0 ns
20 ns 40 ns 60 ns 80 ns 100 ns 120 ns
0 ns
0 ns to 126 ns
wsutton8 : bash
run : vish
ModelSim
Wave

Test Case code Excerpt

```
-- Add your test cases here , use a granularity of 10ns
-- you're going to be defining the input to the barrel shifter and how much
-- then waiting 10ns between each case.
-- TEST CASE 1

--if there is no semicolon, the command continues to the next line.
    input_data <= X"90";
    ctrl_data  <= "011";

--    for i in 0 to 7 loop
--       wait for 10 ns;
--       input_data <= std_logic_vector(unsigned(input_data) + 1);
--       ctrl_data  <= std_logic_vector(unsigned(ctrl_data)
+ 1);
--    end loop;
--    assert false
--    report "Test Case 1 completed"
--    severity failure;
    wait for 10 ns;

---- TEST CASE 2

    input_data <= X"49";
    ctrl_data  <= "110";

--    assert false
--    report "Test Case 2 completed"
--    severity failure;
    wait for 10 ns;
--
--
---- TEST CASE 3

    input_data <= X"47";
    ctrl_data  <= "001";

--    assert false
```

```
--     report "Test Case 3 completed"
--     severity failure;
    wait for 10 ns;
-- between test cases
```

Manual Verification

Since the barrel shifter is essentially bit shifting with a carry function, we can simply move the data bits around. i.e.

0x90=0b10010000 →by 0b010 = 0b00100100 = 0x24
0x49=0b01001001 →by 0b111 = 0b10010010 = 0x92
0x47=0b01000111 →by 0b101 = 0b00111010 = 0x3A

Answer to question in part iv

No I don't think this is the best strategy used. This tests changing input data AND changing control information simultaneously. For a valid test, I would keep the data the same, while toggling the control pins. Then afterwards, I would toggle the data pins, while keeping the control pins the same. This would show full functionality.