CEEN 3130
SWITCHING CIRCUITS THEORY

AUTHOR
Austin Koch

Vending Machine Simulation

PORTFOLIO PROJECT #2
Due May 6th, 2015

The Peter Kiewit Institute
University of Nebraska

Objectives

- **To apply knowledge of VHDL and ASM state machines to a Real Life Problem**
- **To generate code that will function as a realistic Vending Machine**

Design Requirements

Discussion of Constraints:

The purpose of this project is to generate code that will dispense out products based upon user inputs of various coins and selections. This project is an echo of many of the other projects and labs performed throughout the course. This project, though in aims to be a realistic model, will allow some leeway. First of which, the design does not accommodate for continuous inputs and would, in a real life setting, treat a prolonged input as an infinite sum of currency. This was accounted for by user monitored inputs.

The constraints of the project also included the necessity of a multiplexor in lieu of direct input selections, an Adder to attain the running total, a shift register to store that value, and a comparator to signify when enough change has been returned.

Components:

CONTROLLER:

The device uses a controller to input and output signals to and from the components in the lab.

ADDER:

This device asynchronously adds or subtracts two inputs, based upon whether or not a sub line is set high. In this case the storage register contents and the value selected by the mux.

COMPARATOR:

This component compares the value of the storage register with the value of the mux. If the storage is less than the arbitrary value, component will output a high signal indicating the total is too low to return change.

STORAGE REGISTER:

Takes on the value of sum of the Adder. Will store the incoming value if enable storage is high. Sends out stored value to be used by Adder.

System Operation

The machine will monitor to see if any change has been entered into the system, a nickel, dime , or quarter, and will keep a running total of the change deposited. Once the user makes a selection, then the machine will dispense the drink, subtract the cost from the running total, and then return change until there is nothing left in the running total. Change is signified by high outputs R5, R10, and R25, each signifying the value of the coin being changed. The device will signify when enough change has been given when the comparator deems the potential change value is greater than what is left on the store register.

The mux determines its values based on two select lines from the controller. The values change depending on state and will go through a cycle to test and see if using a quarter, dime, or nickel value is most appropriate.

After a price is selected the machine will go through each coin, starting with 25, and use the comparator to determine if the total is less than the change being subtracted. If it is, NEG will go high and the controller will turn off the arithmetic and move onto the next value. Once all three have been used, the total should be zero.

Portfolio Project 2 – Spring 2015.

CEEN 3130 Switching Circuit Theory

In this exercise you will design the electronics for a vending machine. The machine will accept nickels, dimes, and quarters. It has products that cost either 10 cents or 25 cents. It will give the correct change for the amount of money deposited and the product selected.

INPUTS: All inputs will be high for one clock period. They will not change at the same time as the rising edge of the clock. Inputs will be spaced so all of your machine's computations will be done before the next input occurs. You do not have to wait for the input to return to low before proceeding.

N: Nickel has been inserted

D: Dime has been inserted

Q: Quarter has been inserted

P10: Product costing 10 cents has been selected

P25: Product costing 25 cents has been selected

OUTPUTS:

R5: Returns a nickel. Is high for one clock period.

R10: Returns a dime. Is high for one clock period.

R25: Returns a quarter. Is high for one clock period.

OPERATION:

The machine will total the value of coins deposited. Your design does not have to limit the amount deposited. You can assume it will be less than 100.

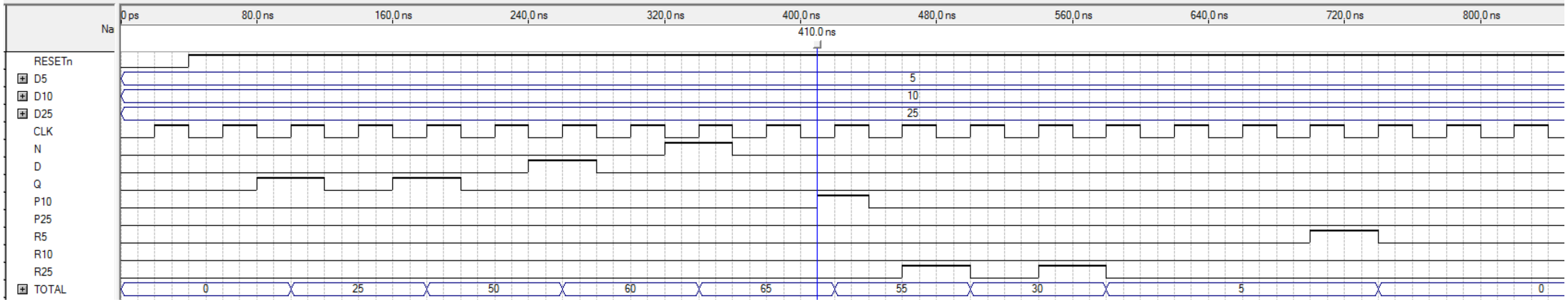
When one of the product selectors is selected the value of the product is deducted from the total. The circuit will then return as many quarters as possible, followed by as many dimes as possible and finally nickels until the total is zero.

All the prior requirements for ASM charts, code, and simulations apply. Simulate using the three input sequences shown. As always, your simulation will have ALL signals into/out of the control unit, states, and values of counters and registers. Show the grouped values of the counters and registers.

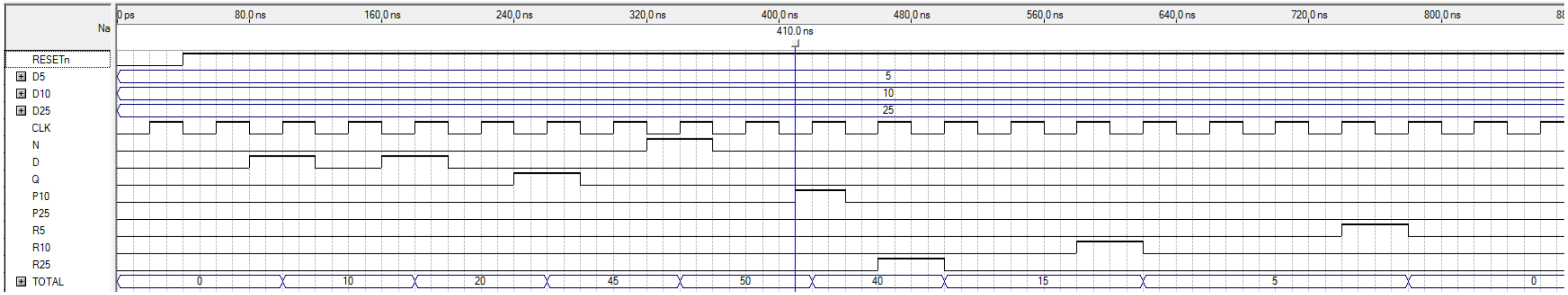
The portfolio project will be submitted in hardcopy and electronic format. For the electronic format, all sections will be assembled into one pdf file with the *name yourlastname_proj2* and sent to Blackboard. (Don't zip the files together) Include the following verification.

I did not receive assistance in designing the circuit other than from the course instructor nor did I share my design with any other student.

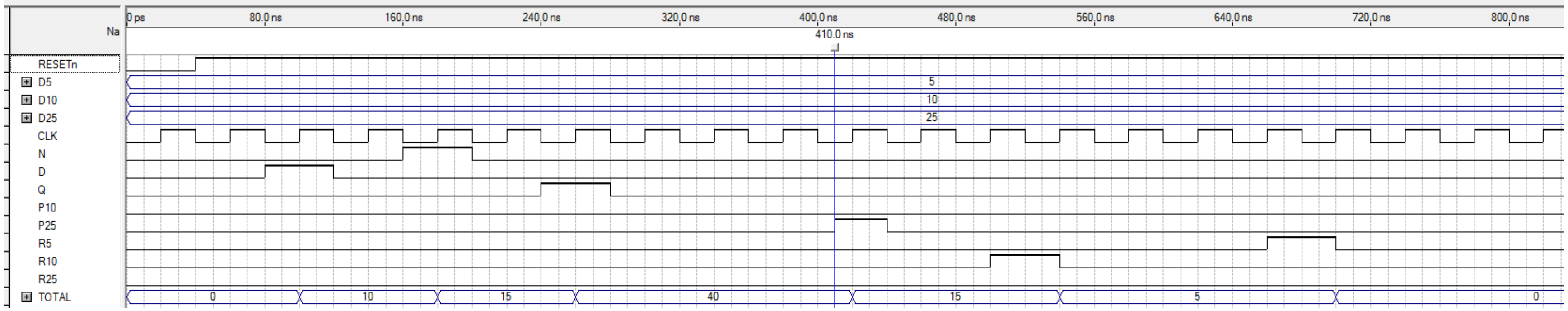
Signed____



Input: Quarter, Quarter, Dime, Nickel = 65; Selection = 10. Change: Quarter, Quarter, Nickel = 55.

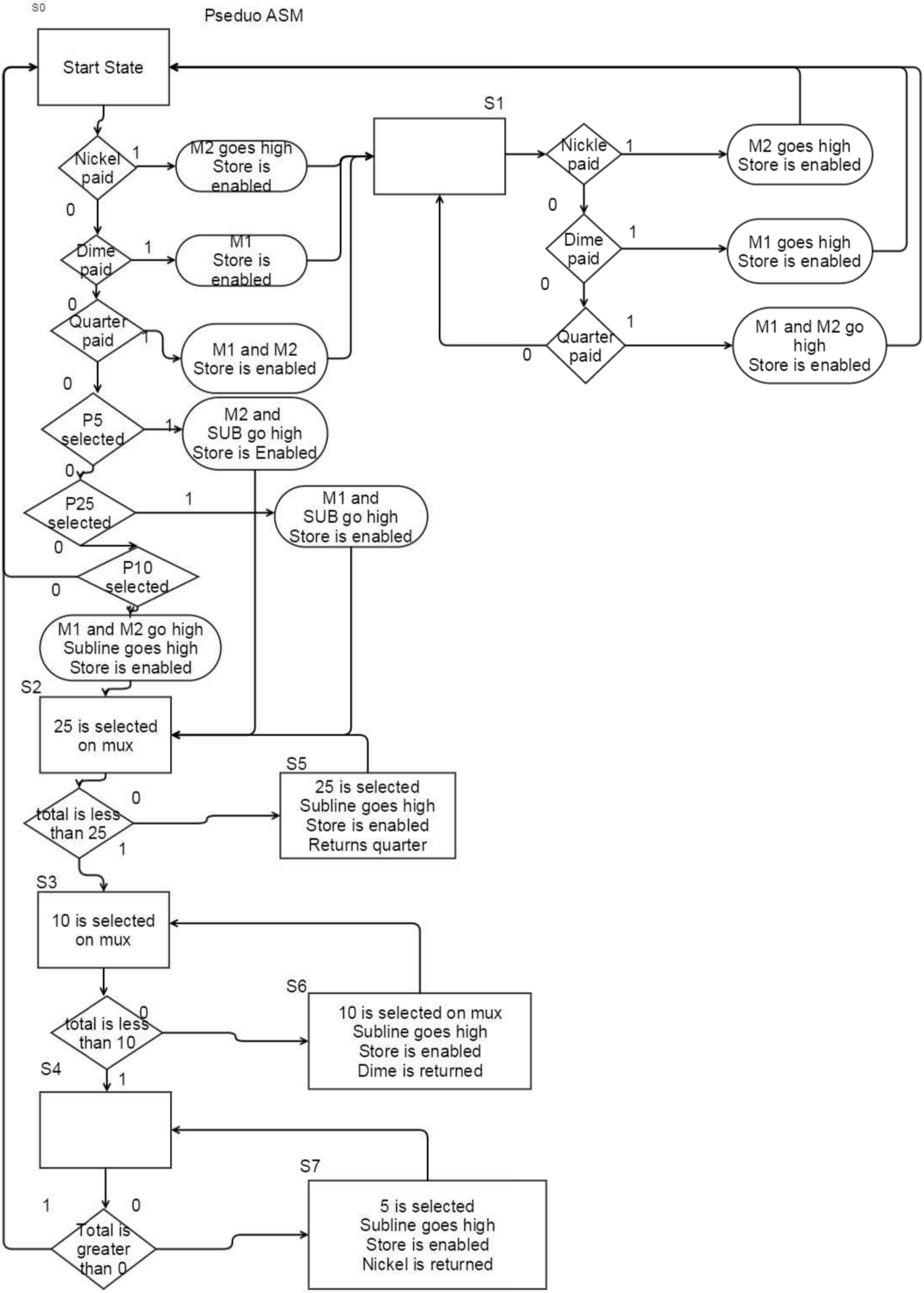


Input: Dime, Dime, Quarter, Nickel = 50; Selection = 10. Change: Quarter, Dime, Nickel = 40.

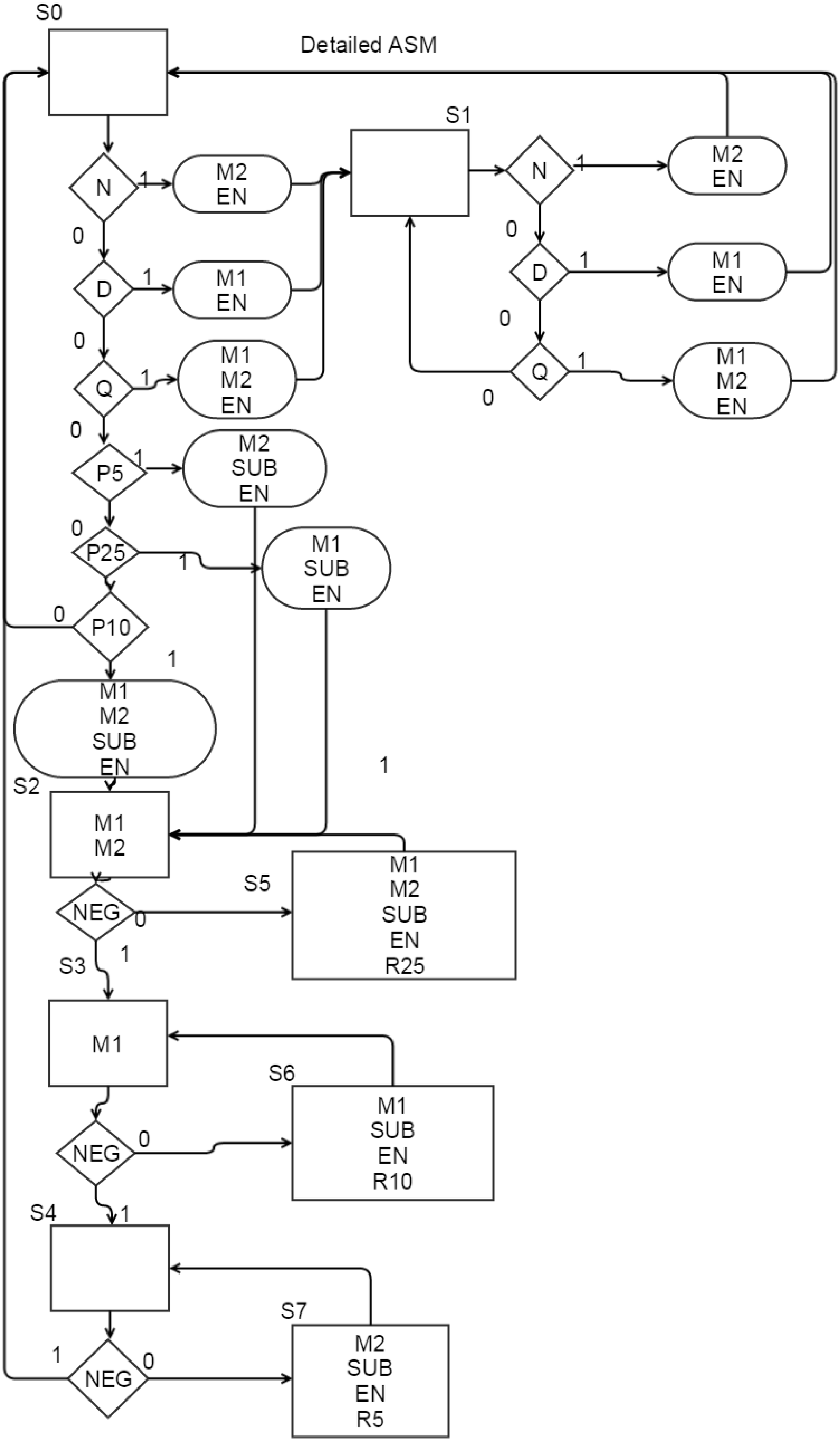


Input: Dime, Nickel, Quarter = 40. Selection = 25. Change: Dime, Nickel = 15.

Pseduo ASM

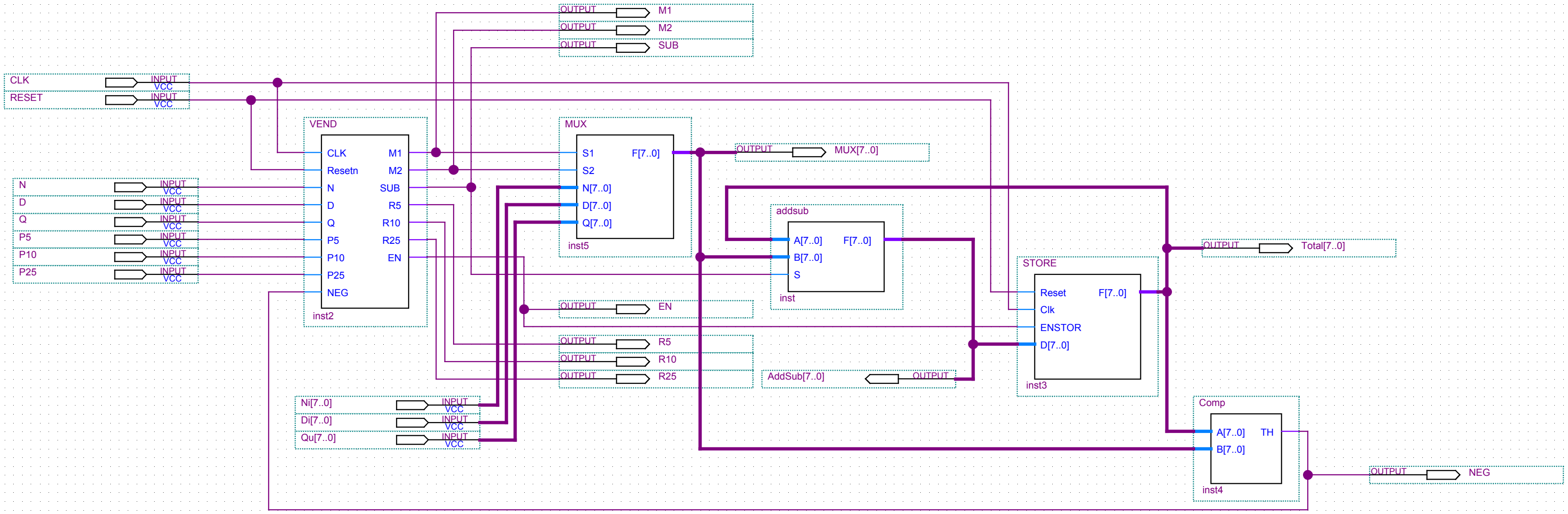

S0
Start State
Nickel paid
1
0
M2 goes high Store is enabled
Dime paid
1
0
M1 Store is enabled
Quarter paid
1
0
M1 and M2 Store is enabled
P5 selected
1
0
M2 and SUB go high Store is Enabled
P25 selected
1
0
M1 and SUB go high Store is enabled
P10 selected
0
M1 and M2 go high Subline goes high Store is enabled
S1
Nickle paid
1
0
M2 goes high Store is enabled
Dime paid
1
0
M1 goes high Store is enabled
Quarter paid
1
0
M1 and M2 go high Store is enabled
S2
25 is selected on mux
total is less than 25
0
1
S5
25 is selected Subline goes high Store is enabled Returns quarter
S3
10 is selected on mux
total is less than 10
0
1
S6
10 is selected on mux Subline goes high Store is enabled Dime is returned
S4
Total is greater than 0
1
0
S7
5 is selected Subline goes high Store is enabled Nickel is returned

Detailed ASM

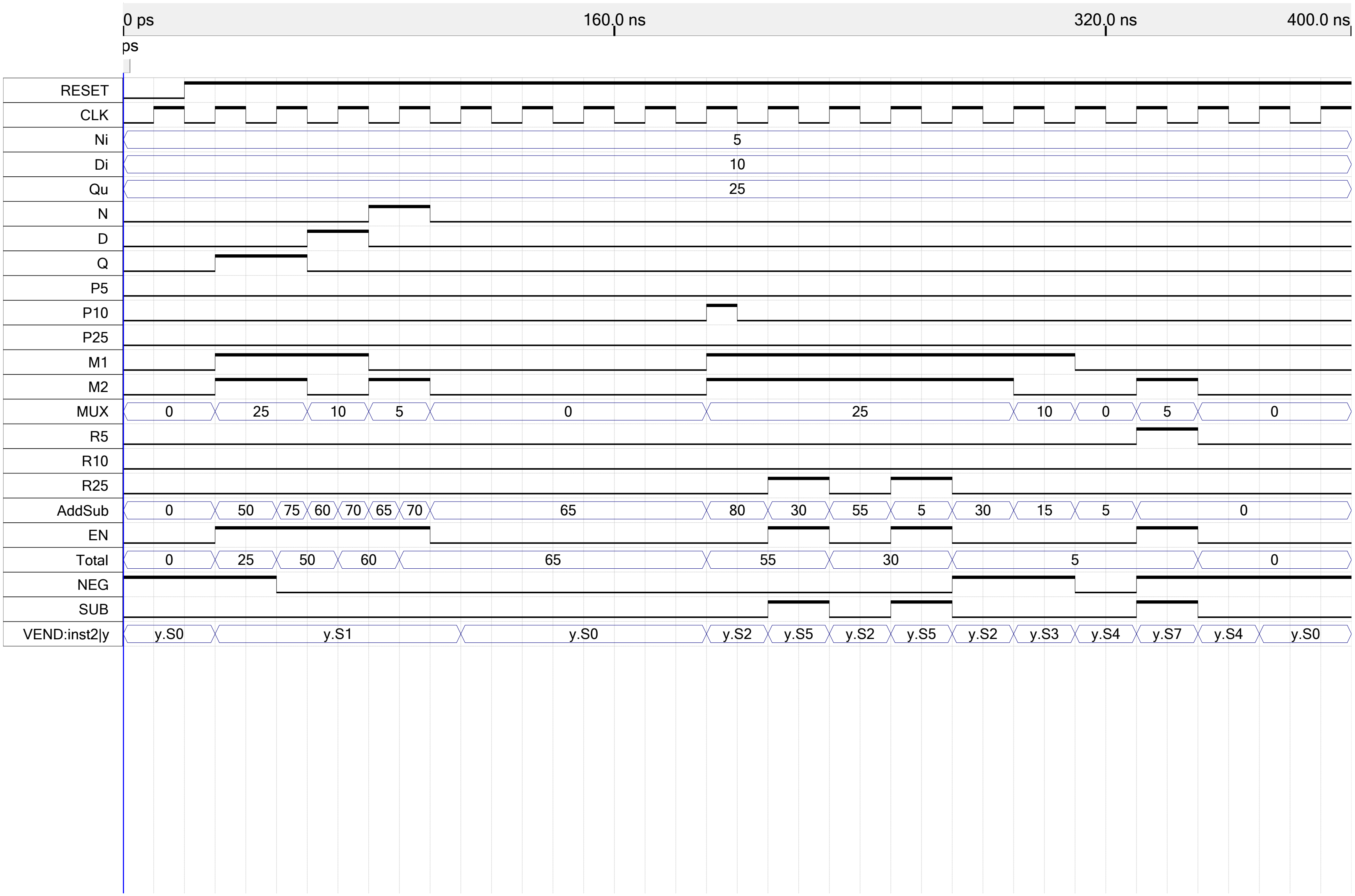


```
--Austin Koch 3130 Portfolio 2--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity addsub is
  port(A, B : in std_logic_vector(7 downto 0);
      S: in std_logic;
      F : out std_logic_vector(7 downto 0));
end addsub;
architecture ADD_SUB of addsub is
  begin
        Process(A, B, S)
        BEGIN
            IF S = '1' THEN
               F <= A - B;
            Else
               F <= A + B;
            END IF;
        END PROCESS;

end ADD_SUB;

------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity MUX is
  port(S1, S2 : IN STD_LOGIC;
      N, D, Q : IN std_logic_vector(7 downto 0);
      F : out std_logic_vector(7 downto 0));

End MUX;

Architecture Behavior of MUX IS
--Signal Fs : STD_LOGIC_VECTOR(7 downto 0);
BEGIN

    Process(N, D, Q, S1, S2)
    BEGIN
        IF S1 = '0' AND S2 = '0' THEN
            F <= "00000000";
        ElsIF S1 = '0' AND S2 = '1' THEN
            F <= N;
        ElsIF S1 = '1' AND S2 = '0' THEN
            F <= D;
        ELSE
            F <= Q;
        END IF;
    END PROCESS;

END Behavior;

```

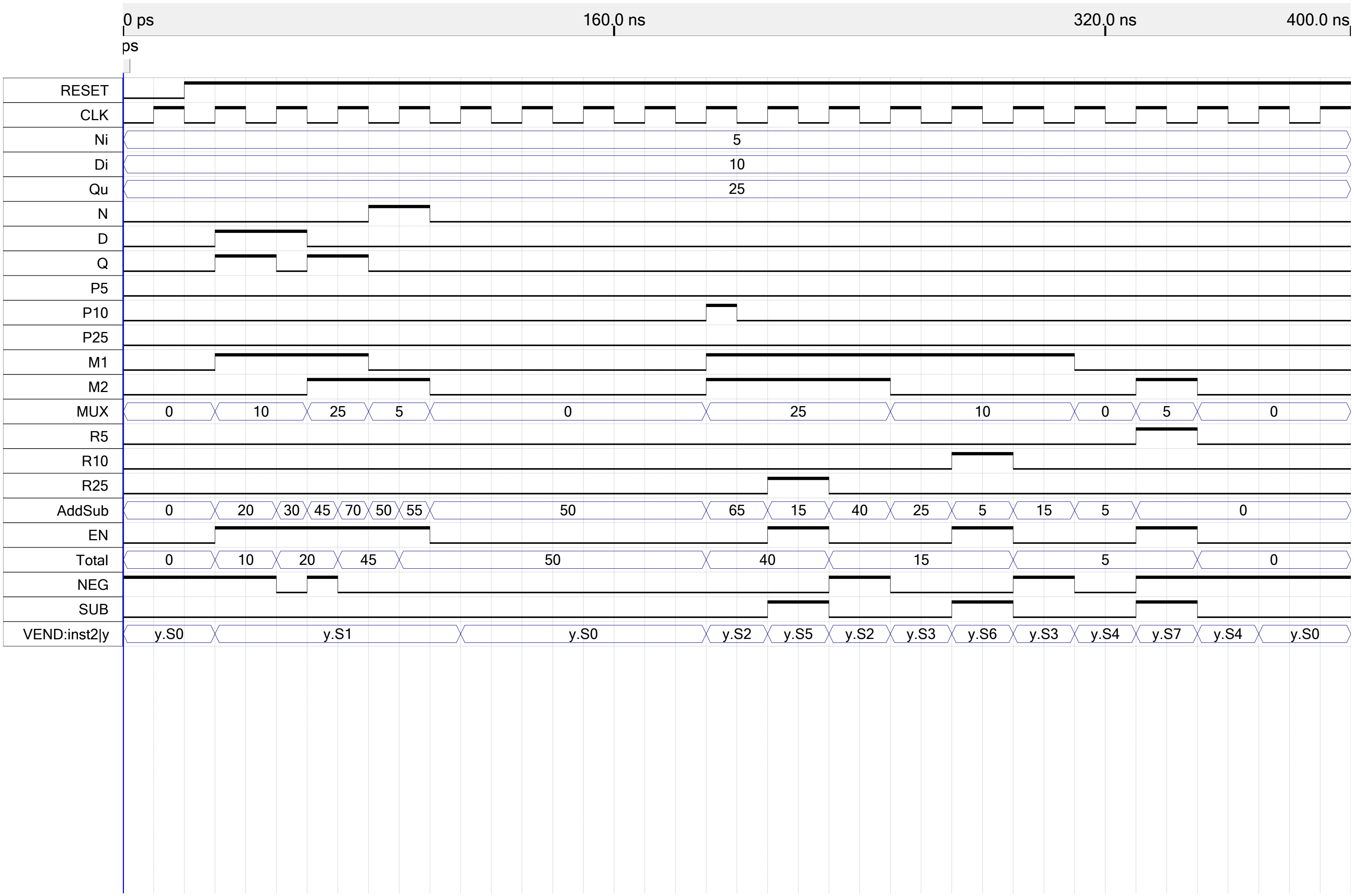
```
---------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

Entity Comp IS
    PORT (A, B : IN STD_Logic_Vector(7 downto 0);
          TH    : OUT STD_LOGIC);
END Comp;

Architecture Behavior OF Comp IS

BEGIN
    Process (A, B)
    BEGIN
        TH <= '1';
        IF  A > B THEN
        TH <= '0';
        END IF;
    END PROCESS;

END Behavior;

---------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity STORE is
  port(Reset, Clk, ENSTOR : IN STD_LOGIC;
      D : in std_logic_vector(7 downto 0);
      F : out std_logic_vector(7 downto 0));
END STORE;

ARCHITECTURE FUNC OF STORE IS
SIGNAL Fs : STD_LOGIC_VECTOR(7 downto 0);
BEGIN
      PROCESS(Reset, CLK, ENSTOR, D, Fs)
      BEGIN
        IF RESET = '0' THEN
            Fs <= "00000000";
        ELSIF CLK'EVENT AND CLK = '1' AND ENSTOR = '1' THEN
            Fs <= D;
        ELSE
            Fs <= Fs;
        END IF;
      END PROCESS;

      F <= Fs;
END FUNC;
------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

ENTITY VEND IS
      PORT( CLK, Resetn, N, D, Q, P5, P10, P25, NEG : IN STD_LOGIC;
            M1, M2, SUB, R5, R10, R25, EN : OUT STD_LOGIC);
END VEND;

ARCHITECTURE BEHAVIOR OF VEND IS
TYPE State_type IS ( S0, S1, S2, S3, S4, S5, S6, S7);
SIGNAL y : State_type;
BEGIN
        Process(CLK, RESETN, y)
        BEGIN
            IF RESETn = '0' THEN
                y <= S0;
            ELSIF CLK'EVENT AND CLK = '1' THEN
                Case y IS
                    WHEN S0 =>
                        IF N = '1' OR D = '1' OR Q = '1' THEN
                            y <= S1;
                        ELSIF P5 = '1' OR P10 = '1' OR P25 = '1' THEN
                            y <= S2;
                        ELSE
                            y <= S0;
                        END IF;
                    WHEN S1 =>
                        IF N = '0' AND D = '0' AND Q = '0' THEN
                            y <= S0;
                        ELSE
                            y <= S1;
                        END IF;
                    WHEN S2 =>
                        IF NEG = '1' THEN
                            y <= S3;
                        ELSE
                            y <= S5;
                        END IF;
                    WHEN S3 =>
                        IF NEG = '1' THEN
                            y <= S4;
                        ELSE
                            y <= S6;
                        END IF;
                    WHEN S4 =>
                        IF NEG = '1' THEN
                            y <= S0;
                        ELSE
                            y <= S7;
                        END IF;
                    WHEN S5 =>
                        y <= S2;
                    WHEN S6 =>
                        y <= S3;
                    WHEN S7 =>
                        y <= S4;
                    WHEN OTHERS =>
                        y <= S0;
                END CASE;
            END IF;
            END PROCESS;


        Process(y, P5, P10, P25, CLK, N, D, Q, NEG)
        BEGIN
            M1 <= '0';
            M2 <= '0';
            SUB <= '0';
            EN <= '0';
            R5 <= '0';
            R10 <= '0';
            R25 <= '0';


                case y IS
                    WHEN S0 =>
                        IF N = '1' THEN
                            M2 <= '1';
                            EN <= '1';
                        ELSIF D = '1' THEN
                            M1 <= '1';
                            EN <= '1';
                        ELSIF Q = '1' THEN
                            M1 <= '1';
                            M2 <= '1';
                            EN <= '1';
                        ELSIF P5 = '1' THEN
                            SUB <= '1';
                            M2 <= '1';
                            EN <= '1';
                        ELSIF P10 = '1' THEN
                            SUB <= '1';
                            M1 <= '1';
                            EN <= '1';
                        ELSIF P25 = '1' THEN
                            SUB <= '1';
                            M1 <= '1';
                            M2 <= '1';
                            EN <= '1';
                        END IF;
                    WHEN S1 =>
                        IF N = '1' THEN
                            M2 <= '1';
                            EN <= '1';
                        ELSIF D = '1' THEN
                            M1 <= '1';
                            EN <= '1';
                        ELSIF Q = '1' THEN
                            M1 <= '1';
                            M2 <= '1';
                            EN <= '1';
                        END IF;
                    WHEN S2 =>
                        M1 <= '1';
                        M2 <= '1';
                    WHEN S3 =>
                        M1 <= '1';
                    WHEN S4 =>
                    WHEN S5 =>
                        M1 <= '1';
                        M2 <= '1';
                        SUB <= '1';
                        EN <= '1';
                        R25 <= '1';
                    WHEN S6 =>
                        M1 <= '1';
                        SUB <= '1';
                        EN <= '1';
                        R10 <= '1';
                    WHEN S7 =>
                        M2 <= '1';
                        SUB <= '1';
                        EN <= '1';
                        R5 <= '1';
                END CASE;
            END PROCESS;

END BEHAVIOR;

```

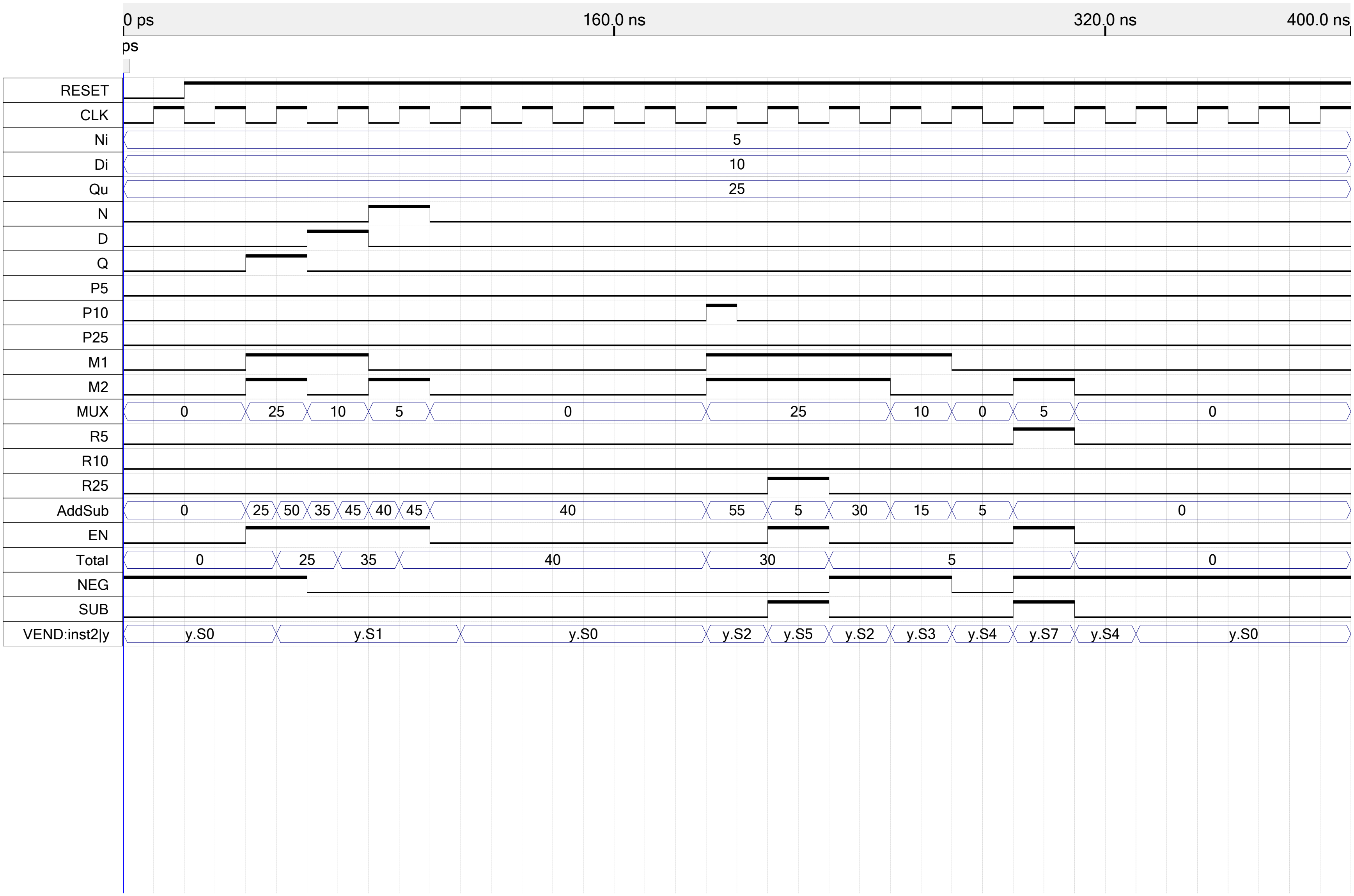

OUTPUT
M1
OUTPUT
M2
OUTPUT
SUB
CLK
INPUT
VCC
RESET
INPUT
VCC
VEND
CLK
Resetn
N
D
Q
P5
P10
P25
NEG
M1
M2
SUB
R5
R10
R25
EN
inst2
N
D
Q
P5
P10
P25
INPUT
VCC
MUX
S1
S2
N[7..0]
D[7..0]
Q[7..0]
F[7..0]
inst5
OUTPUT
MUX[7..0]
addsub
A[7..0]
B[7..0]
S
F[7..0]
inst
OUTPUT
Total[7..0]
STORE
Reset
Clk
ENSTOR
D[7..0]
F[7..0]
inst3
OUTPUT
EN
OUTPUT
R5
OUTPUT
R10
OUTPUT
R25
AddSub[7..0]
OUTPUT
Ni[7..0]
Di[7..0]
Qu[7..0]
INPUT
VCC
Comp
A[7..0]
B[7..0]
TH
inst4
OUTPUT
NEG

Date: May 06, 2015 db/PorfolioFInal.sim.cvwf Project: PorfolioFInal



Date: May 06, 2015 db/PorfolioFInal.sim.cvwf* Project: PorfolioFInal




0 ps
160.0 ns
320.0 ns
400.0 ns
ps
RESET
CLK
Ni 5
Di 10
Qu 25
N
D
Q
P5
P10
P25
M1
M2
MUX 0 25 10 5 0 25 10 0 5 0
R5
R10
R25
AddSub 0 25 50 35 45 40 45 40 55 5 30 15 5 0
EN
Total 0 25 35 40 30 5 0
NEG
SUB
VEND:inst2|y y.S0 y.S1 y.S0 y.S2 y.S5 y.S2 y.S3 y.S4 y.S7 y.S4 y.S0