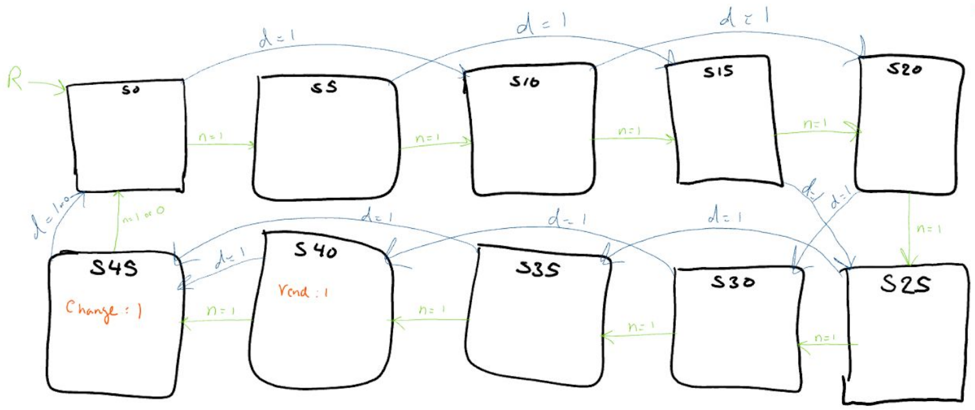
State Diagram 1

14.20



* if all inputs are 0 then remain in Current state unless in final state

Vending Machine 1

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


entity VM is--14.20
	port(clk, reset, nickle, dime : in std_logic;
		vend, change : out std_logic);
end;

architecture beh of VM is

	type state is (S0,S5,S10,S15,S20,S25,S30,S35,S40,S45);
	signal cs,ns : state;
	signal vendtmp, changetmp: std_logic;

	begin
	process(clk,reset)
		begin
			if(reset = '1') then
				cs<=S0;
			elsif(clk'event and clk='1') then
				cs<=ns;
			end if;
		end process;
```

```
process(nickle,dime,cs)
        begin
                case cs is
                        when S0 =>
                                if (nickle = '0' and dime = '0') then
                                        ns <= cs ;
                                elsif(nickle = '1' and dime = '0') then
                                        ns <= S5;
                                elsif(nickle = '0' and dime = '1') then
                                        ns <= S10;
                                end if;
                                        vendtmp <= '0';
                                        changetmp <= '0';
                        when S5 =>
                                if (nickle = '0' and dime = '0') then
                                        ns <= cs ;
                                elsif(nickle = '1' and dime = '0') then
                                        ns <= S10;
                                elsif(nickle = '0' and dime = '1') then
                                        ns <= S15;
                                end if;
                                        vendtmp <= '0';
                                        changetmp <= '0';

                        when S10 =>
                                if (nickle = '0' and dime = '0') then
                                        ns <= cs ;
                                elsif(nickle = '1' and dime = '0') then
                                        ns <= S15;
                                elsif(nickle = '0' and dime = '1') then
                                        ns <= S20;
                                end if;
                                        vendtmp <= '0';
                                        changetmp <= '0';
                        when S15 =>
                                if (nickle = '0' and dime = '0') then
                                        ns <= cs ;
                                elsif(nickle = '1' and dime = '0') then
                                        ns <= S20;
                                elsif(nickle = '0' and dime = '1') then
                                        ns <= S25;
                                end if;
                                        vendtmp <= '0';
        changetmp <= '0';
when S20 =>
    if (nickle = '0' and dime = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0') then
        ns <= S25;
    elsif(nickle = '0' and dime = '1') then
        ns <= S30;
    end if;
        vendtmp <= '0';
        changetmp <= '0';
when S25 =>
    if (nickle = '0' and dime = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0') then
        ns <= S30;
    elsif(nickle = '0' and dime = '1') then
        ns <= S35;
    end if;
        vendtmp <= '0';
        changetmp <= '0';
when S30 =>
    if (nickle = '0' and dime = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0') then
        ns <= S35;
    elsif(nickle = '0' and dime = '1') then
        ns <= S40;
    end if;
        vendtmp <= '0';
        changetmp <= '0';
when S35 =>
    if (nickle = '0' and dime = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0') then
        ns <= S40;
    elsif(nickle = '0' and dime = '1') then
        ns <= S45;
    end if;
        vendtmp <= '0';
        changetmp <= '0';
when S40 =>
    if (nickle = '0' and dime = '0') then
```

```
                        ns <= cs ;
                    elsif(nickle = '1' and dime = '0') then
                        ns <= S45;
                    elsif(nickle = '0' and dime = '1') then
                        ns <= S45; --though the total would now be 50
cents, S45 will handle a despensing of change
                    end if;
                        vendtmp <= '1';
                        changetmp <= '0';
                when S45 =>
                    if (nickle = '0' and dime = '0') then
                        ns <= S0 ;
                    elsif(nickle = '1' and dime = '0') then
                        ns <= S0 ;
                    elsif(nickle = '0' and dime = '1') then
                        ns <= S0;
                    end if;
                        vendtmp <= '0';
                        changetmp <= '1';
            end case;

    end process;

        vend <= vendtmp;
        change <= changetmp;

end beh;
```

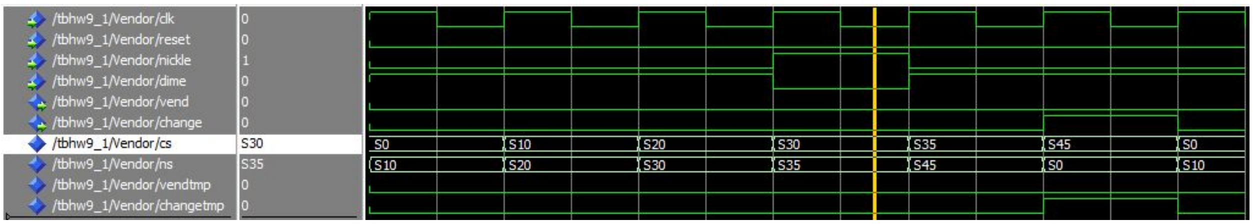
Test Bench 1

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

entity TBHW9_1 is
end;

architecture beh of TBHW9_1 is
        --Component Declaration
        component VM is
                port(clk, reset, nickle, dime : in std_logic;
                                vend, change : out std_logic);
        end component;

--input signals
signal clk, reset, nickle, dime : std_logic;
--output signals
signal vend, change : std_logic;


begin
        Vendor : VM port map(clk,reset,nickle,dime,vend,change);

sim : process

begin
        clk<='1';
        reset<='0';
        nickle<='0';
        dime<='1';
        wait for 100ns;
        --10c

        clk<='0';--low
        reset<='0';
        nickle<='0';
        dime<='1';
        wait for 100ns;

        clk<='1';
        reset<='0';
```

```
nickle<='0';
dime<='1';
wait for 100ns;
--20c

clk<='0';--low
reset<='0';
nickle<='0';
dime<='1';
wait for 100ns;

clk<='1';
reset<='0';
nickle<='0';
dime<='1';
wait for 100ns;
--30c

clk<='0';--low
reset<='0';
nickle<='0';
dime<='1';
wait for 100ns;

clk<='1';
reset<='0';
nickle<='1';
dime<='0';
wait for 100ns;
--35c

clk<='0';--low
reset<='0';
nickle<='1';
dime<='0';
wait for 100ns;

clk<='1';
reset<='0';
nickle<='0';
dime<='1';
wait for 100ns;
--45c
```

```
        clk<='0';--low
        reset<='0';
        nickle<='0';
        dime<='1';
        wait for 100ns;

end process;
end beh;
```

Simulation 1

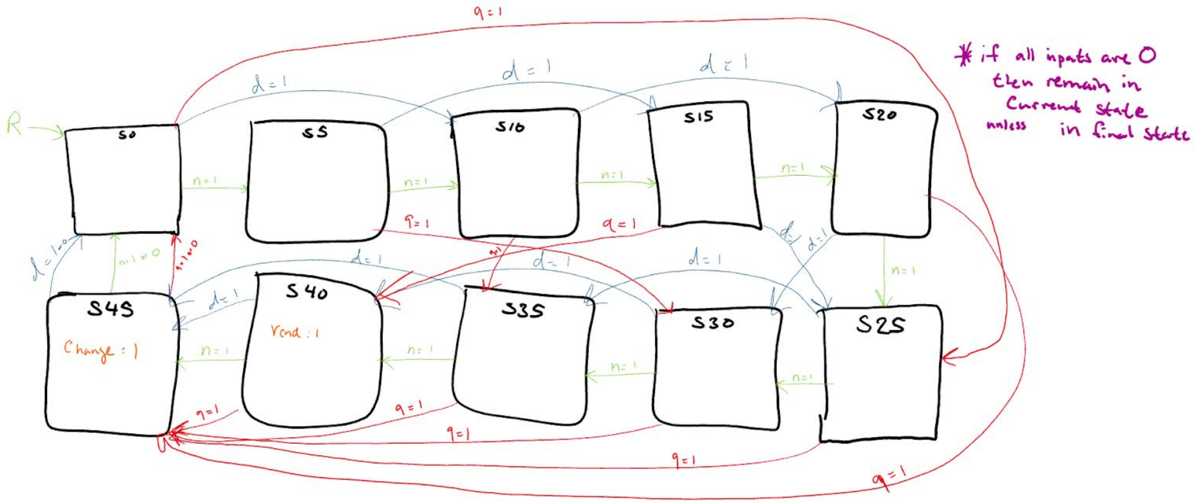


```
/tbhw9_1/Vendor/clk         0
/tbhw9_1/Vendor/reset       0
/tbhw9_1/Vendor/nickle      1
/tbhw9_1/Vendor/dime        0
/tbhw9_1/Vendor/vend        0
/tbhw9_1/Vendor/change      0
/tbhw9_1/Vendor/cs          S30    S0   S10  S20  S30  S35  S45  S0
/tbhw9_1/Vendor/ns          S35    S10  S20  S30  S35  S45  S0   S10
/tbhw9_1/Vendor/vendtmp     0
/tbhw9_1/Vendor/changetmp   0
```

Description 1: Here we have 4 inputs : clock, reset, nickel, dime. 2 outputs vend and change. What i did in my code and simulation was showed how inserting the money allows for the state to traverse through the state and eventually vend the item to the user once the right amount of money is inserted. Also if the user inserts more money than is needed then the vending machine will dispense change then reset the amount (or state) to 0.

State Diagram 2

14.23



*if all inputs are 0 then remain in Current state unless in final state

Vending Machine 2

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



entity VM2 is--14.23
          port(clk, reset, nickle, dime, quarter : in std_logic;
                    vend, change : out std_logic);
end;

architecture beh of VM2 is

          type state is (S0,S5,S10,S15,S20,S25,S30,S35,S40,S45);
          signal cs,ns : state;
          signal vendtmp, changetmp: std_logic;

          begin
          process(clk,reset)
                    begin
                              if(reset = '1') then
                                        cs<=S0;
                              elsif(clk'event and clk='1') then
                                        cs<=ns;
```

```
        end if;
    end process;
process(nickle,dime,cs)
    begin
        case cs is
            when S0 =>
                if (nickle = '0' and dime = '0' and quarter = '0') then
                    ns <= cs ;
                elsif(nickle = '1' and dime = '0' and quarter = '0') then
                    ns <= S5;
                elsif(nickle = '0' and dime = '1' and quarter = '0') then
                    ns <= S10;
                elsif(nickle = '0' and dime = '0' and quarter = '1') then
                    ns<= S25;
                end if;
                    vendtmp <= '0';
                    changetmp <= '0';
            when S5 =>
                if (nickle = '0' and dime = '0' and quarter = '0') then
                    ns <= cs ;
                elsif(nickle = '1' and dime = '0' and quarter = '0') then
                    ns <= S10;
                elsif(nickle = '0' and dime = '1' and quarter = '0') then
                    ns <= S15;
                elsif(nickle = '0' and dime = '0' and quarter = '1') then
                    ns<= S30;
                end if;
                    vendtmp <= '0';
                    changetmp <= '0';

            when S10 =>
                if (nickle = '0' and dime = '0' and quarter = '0') then
                    ns <= cs ;
                elsif(nickle = '1' and dime = '0' and quarter = '0') then
                    ns <= S15;
                elsif(nickle = '0' and dime = '1' and quarter = '0') then
                    ns <= S20;
                elsif(nickle = '0' and dime = '0' and quarter = '1') then
                    ns<= S35;
                end if;
                    vendtmp <= '0';
                    changetmp <= '0';
            when S15 =>
    if (nickle = '0' and dime = '0' and quarter = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0' and quarter = '0') then
        ns <= S20;
    elsif(nickle = '0' and dime = '1' and quarter = '0') then
        ns <= S25;
    elsif(nickle = '0' and dime = '0' and quarter = '1') then
        ns<= S40;
    end if;
        vendtmp <= '0';
        changetmp <= '0';
when S20 =>
    if (nickle = '0' and dime = '0' and quarter = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0' and quarter = '0') then
        ns <= S25;
    elsif(nickle = '0' and dime = '1' and quarter = '0') then
        ns <= S30;
    elsif(nickle = '0' and dime = '0' and quarter = '1') then
        ns<= S45;
    end if;
        vendtmp <= '0';
        changetmp <= '0';
when S25 =>
    if (nickle = '0' and dime = '0' and quarter = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0' and quarter = '0') then
        ns <= S30;
    elsif(nickle = '0' and dime = '1' and quarter = '0') then
        ns <= S35;
    elsif(nickle = '0' and dime = '0' and quarter = '1') then
        ns<= S45;
    end if;
        vendtmp <= '0';
        changetmp <= '0';
when S30 =>
    if (nickle = '0' and dime = '0' and quarter = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0' and quarter = '0') then
        ns <= S35;
    elsif(nickle = '0' and dime = '1' and quarter = '0') then
        ns <= S40;
    elsif(nickle = '0' and dime = '0' and quarter = '1') then
```

```
                    ns<= S45;
                end if;
                    vendtmp <= '0';
                    changetmp <= '0';
            when S35 =>
                if (nickle = '0' and dime = '0' and quarter = '0') then
                    ns <= cs ;
                elsif(nickle = '1' and dime = '0' and quarter = '0') then
                    ns <= S40;
                elsif(nickle = '0' and dime = '1' and quarter = '0') then
                    ns <= S45;
                elsif(nickle = '0' and dime = '0' and quarter = '1') then
                    ns<= S45;
                end if;
                    vendtmp <= '0';
                    changetmp <= '0';
            when S40 =>--vend state
                if (nickle = '0' and dime = '0' and quarter = '0') then
                    ns <= cs ;
                elsif(nickle = '1' and dime = '0' and quarter = '0') then
                    ns <= S45;
                elsif(nickle = '0' and dime = '1' and quarter = '0') then
                    ns <= S45;
                elsif(nickle = '0' and dime = '0' and quarter = '1') then
                    ns<= S45;
                end if;
                    vendtmp <= '1';
                    changetmp <= '0';
            when S45 =>--change state
                if (nickle = '0' and dime = '0' and quarter = '0') then
                    ns <= cs ;
                elsif(nickle = '1' and dime = '0' and quarter = '0') then
                    ns <= S0;
                elsif(nickle = '0' and dime = '1' and quarter = '0') then
                    ns <= S0;
                elsif(nickle = '0' and dime = '0' and quarter = '1') then
                    ns<= S0;
                end if;

                    vendtmp <= '0';
                    changetmp <= '1';
    end case;
end process;
```

```
                vend <= vendtmp;
                change <= changetmp;

end beh;
```

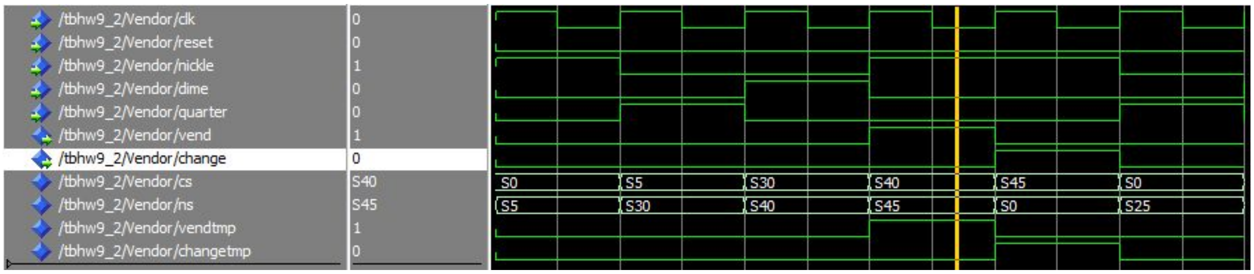
Test Bench 2

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

entity TBHW9_2 is
end;

architecture beh of TBHW9_2 is
        --Component Declaration
        component VM2 is
                port(clk, reset, nickle, dime, quarter : in std_logic;
                                vend, change : out std_logic);
        end component;

--input signals
signal clk, reset, nickle, dime, quarter : std_logic;
--output signals
signal vend, change : std_logic;


begin
        Vendor : VM2 port map(clk,reset,nickle,dime,quarter,vend,change);

sim : process

begin
```

```
clk<='1';
reset<='0';
nickle<='1';
dime<='0';
quarter<='0';
wait for 100ns;
--5c

clk<='0';--low
reset<='0';
nickle<='1';
dime<='0';
quarter<='0';
wait for 100ns;

clk<='1';
reset<='0';
nickle<='0';
dime<='0';
quarter<='1';
wait for 100ns;
--30c

clk<='0';--low
reset<='0';
nickle<='0';
dime<='0';
quarter<='1';
wait for 100ns;

clk<='1';
reset<='0';
nickle<='0';
dime<='1';
quarter<='0';
wait for 100ns;
--40c

clk<='0';--low
reset<='0';
nickle<='0';
dime<='1';
quarter<='0';
```

```
        wait for 100ns;

        clk<='1';
        reset<='0';
        nickle<='1';
        dime<='0';
        wait for 100ns;
        --45c

        clk<='0';--low
        reset<='0';
        nickle<='1';
        dime<='0';
        wait for 100ns;


end process;
end beh;
```

Simulation 2

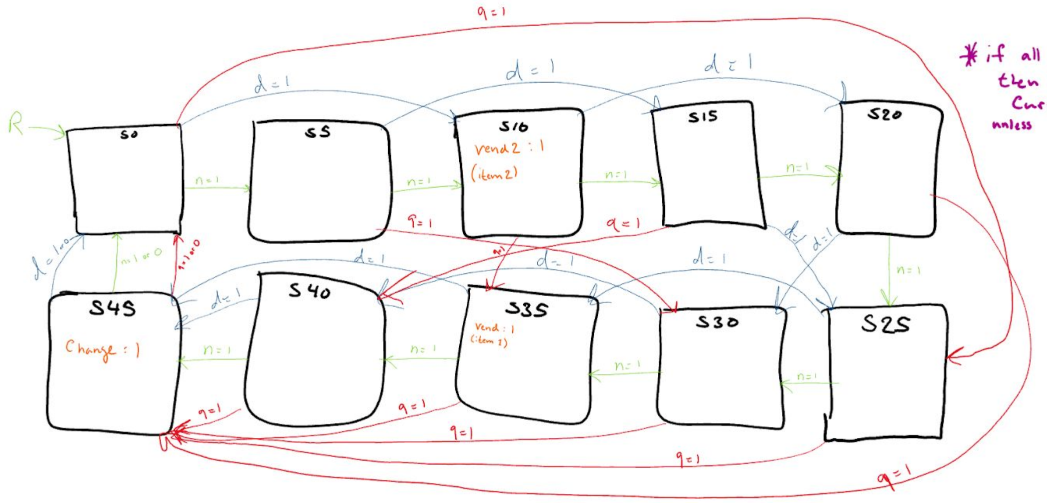


```
/tbhw9_2/Vendor/clk         0
/tbhw9_2/Vendor/reset       0
/tbhw9_2/Vendor/nickle      1
/tbhw9_2/Vendor/dime        0
/tbhw9_2/Vendor/quarter     0
/tbhw9_2/Vendor/vend        1
/tbhw9_2/Vendor/change      0
/tbhw9_2/Vendor/cs          S40
/tbhw9_2/Vendor/ns          S45
/tbhw9_2/Vendor/vendtmp     1
/tbhw9_2/Vendor/changetmp   0
```

Description 2: Here we have 5 inputs : clock, reset, nickel, dime, and quarter, and 2 outputs vend and change. What i did in my code and simulation was showed how inserting the money allows for the state to traverse through the state and eventually vend the item to the user once the right amount of money is inserted. Also if the user inserts more money than is needed then the vending machine will dispense change then reset the amount (or state) to 0.

State Diagram 3

14.23(edited)



*if all inputs are 0
then remain in
Current state
unless in final state

Vending Machine 3

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


entity VM3 is--14.23 (added a second item and changed the price of one of the items)
        port(clk, reset, nickle, dime, quarter : in std_logic;
                vend, vend2, change : out std_logic);
end;

architecture beh of VM3 is

        type state is (S0,S5,S10,S15,S20,S25,S30,S35,S40,S45);
        signal cs,ns : state;
        signal vendtmp, vend2tmp, changetmp: std_logic;

        begin
        process(clk,reset)
                begin
                        if(reset = '1') then
                                cs<=S0;
        elsif(clk'event and clk='1') then
            cs<=ns;
        end if;
    end process;
process(nickle,dime,cs)
    begin
        case cs is
            when S0 =>
                if (nickle = '0' and dime = '0' and quarter = '0') then
                    ns <= cs ;
                elsif(nickle = '1' and dime = '0' and quarter = '0') then
                    ns <= S5;
                elsif(nickle = '0' and dime = '1' and quarter = '0') then
                    ns <= S10;
                elsif(nickle = '0' and dime = '0' and quarter = '1') then
                    ns<= S25;
                end if;
                    vendtmp <= '0';
                    vend2tmp <= '0';
                    changetmp <= '0';
            when S5 =>
                if (nickle = '0' and dime = '0' and quarter = '0') then
                    ns <= cs ;
                elsif(nickle = '1' and dime = '0' and quarter = '0') then
                    ns <= S10;
                elsif(nickle = '0' and dime = '1' and quarter = '0') then
                    ns <= S15;
                elsif(nickle = '0' and dime = '0' and quarter = '1') then
                    ns<= S30;
                end if;
                    vendtmp <= '0';
                    vend2tmp <= '0';
                    changetmp <= '0';

            when S10 =>
                if (nickle = '0' and dime = '0' and quarter = '0') then
                    ns <= cs ;
                elsif(nickle = '1' and dime = '0' and quarter = '0') then
                    ns <= S15;
                elsif(nickle = '0' and dime = '1' and quarter = '0') then
                    ns <= S20;
                elsif(nickle = '0' and dime = '0' and quarter = '1') then
                    ns<= S35;
    end if;
        vendtmp <= '0';
        vend2tmp <= '1';--vend item 2
        changetmp <= '0';
when S15 =>
    if (nickle = '0' and dime = '0' and quarter = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0' and quarter = '0') then
        ns <= S20;
    elsif(nickle = '0' and dime = '1' and quarter = '0') then
        ns <= S25;
    elsif(nickle = '0' and dime = '0' and quarter = '1') then
        ns<= S40;
    end if;
        vendtmp <= '0';
        vend2tmp <= '0';
        changetmp <= '0';
when S20 =>
    if (nickle = '0' and dime = '0' and quarter = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0' and quarter = '0') then
        ns <= S25;
    elsif(nickle = '0' and dime = '1' and quarter = '0') then
        ns <= S30;
    elsif(nickle = '0' and dime = '0' and quarter = '1') then
        ns<= S45;
    end if;
        vendtmp <= '0';
        vend2tmp <= '0';
        changetmp <= '0';
when S25 =>
    if (nickle = '0' and dime = '0' and quarter = '0') then
        ns <= cs ;
    elsif(nickle = '1' and dime = '0' and quarter = '0') then
        ns <= S30;
    elsif(nickle = '0' and dime = '1' and quarter = '0') then
        ns <= S35;
    elsif(nickle = '0' and dime = '0' and quarter = '1') then
        ns<= S45;
    end if;
        vendtmp <= '0';
        vend2tmp <= '0';
        changetmp <= '0';
```

```
when S30 =>
        if (nickle = '0' and dime = '0' and quarter = '0') then
                ns <= cs ;
        elsif(nickle = '1' and dime = '0' and quarter = '0') then
                ns <= S35;
        elsif(nickle = '0' and dime = '1' and quarter = '0') then
                ns <= S40;
        elsif(nickle = '0' and dime = '0' and quarter = '1') then
                ns<= S45;
        end if;
                vendtmp <= '0';
                vend2tmp <= '0';
                changetmp <= '0';
when S35 =>
        if (nickle = '0' and dime = '0' and quarter = '0') then
                ns <= cs ;
        elsif(nickle = '1' and dime = '0' and quarter = '0') then
                ns <= S40;
        elsif(nickle = '0' and dime = '1' and quarter = '0') then
                ns <= S45;
        elsif(nickle = '0' and dime = '0' and quarter = '1') then
                ns<= S45;
        end if;
                vendtmp <= '1';--vend item 1
                vend2tmp <= '0';
                changetmp <= '0';
when S40 =>--vend state
        if (nickle = '0' and dime = '0' and quarter = '0') then
                ns <= cs ;
        elsif(nickle = '1' and dime = '0' and quarter = '0') then
                ns <= S45;
        elsif(nickle = '0' and dime = '1' and quarter = '0') then
                ns <= S45;
        elsif(nickle = '0' and dime = '0' and quarter = '1') then
                ns<= S45;
        end if;
                vendtmp <= '0';
                vend2tmp <= '0';
                changetmp <= '0';
when S45 =>--change state
        if (nickle = '0' and dime = '0' and quarter = '0') then
                ns <= cs ;
        elsif(nickle = '1' and dime = '0' and quarter = '0') then
```

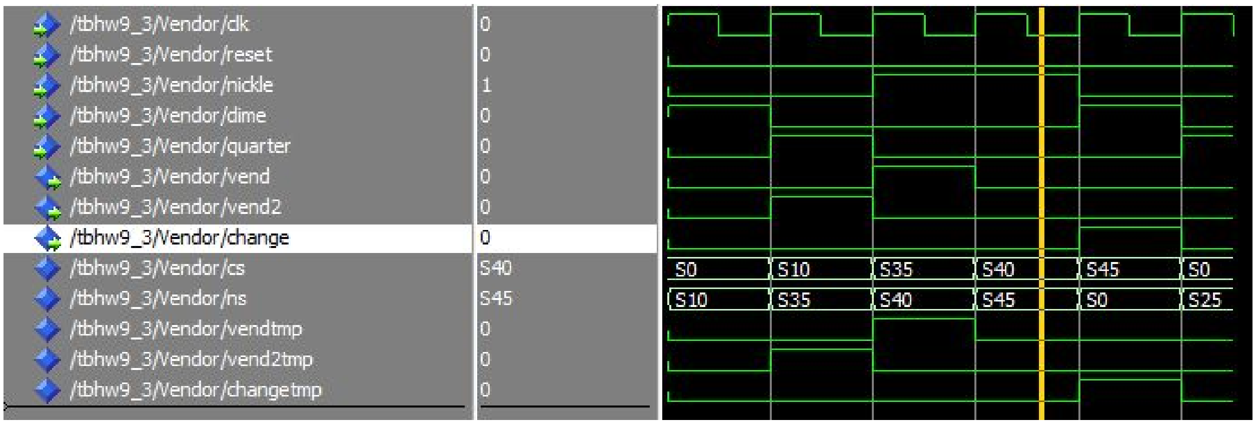
```
                        ns <= S0;
                    elsif(nickle = '0' and dime = '1' and quarter = '0') then
                        ns <= S0;
                    elsif(nickle = '0' and dime = '0' and quarter = '1') then
                        ns<= S0;
                    end if;

                        vendtmp <= '0';
                        vend2tmp <= '0';
                        changetmp <= '1';
        end case;
    end process;

        vend <= vendtmp;
        vend2 <= vend2tmp;
        change <= changetmp;

end beh;
```

Test Bench 3

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

entity TBHW9_3 is
end;

architecture beh of TBHW9_3 is
    --Component Declaration
    component VM3 is
        port(clk, reset, nickle, dime, quarter : in std_logic;
                vend, vend2, change : out std_logic);
    end component;

--input signals
signal clk, reset, nickle, dime, quarter : std_logic;
--output signals
signal vend, vend2, change : std_logic;


begin
    Vendor : VM3 port map(clk,reset,nickle,dime,quarter,vend,vend2,change);
```

```
sim : process

begin
        clk<='1';
        reset<='0';
        nickle<='0';
        dime<='1';
        quarter<='0';
        wait for 100ns;
        --10c

        clk<='0';--low
        reset<='0';
        nickle<='0';
        dime<='1';
        quarter<='0';
        wait for 100ns;

        clk<='1';
        reset<='0';
        nickle<='0';
        dime<='0';
        quarter<='1';
        wait for 100ns;
        --35c

        clk<='0';--low
        reset<='0';
        nickle<='0';
        dime<='0';
        quarter<='1';
        wait for 100ns;

        clk<='1';
        reset<='0';
        nickle<='1';
        dime<='0';
        quarter<='0';
        wait for 100ns;
        --40c

        clk<='0';--low
        reset<='0';
        nickle<='1';
        dime<='0';
        quarter<='0';
        wait for 100ns;

        clk<='1';
        reset<='0';
        nickle<='1';
        dime<='0';
        wait for 100ns;
        --45c

        clk<='0';--low
        reset<='0';
        nickle<='1';
        dime<='0';
        wait for 100ns;


end process;
end beh;
```

Simulation 3



```
/tbhw9_3/Vendor/clk         0
/tbhw9_3/Vendor/reset       0
/tbhw9_3/Vendor/nickle      1
/tbhw9_3/Vendor/dime        0
/tbhw9_3/Vendor/quarter     0
/tbhw9_3/Vendor/vend        0
/tbhw9_3/Vendor/vend2       0
/tbhw9_3/Vendor/change      0
/tbhw9_3/Vendor/cs          S40
/tbhw9_3/Vendor/ns          S45
/tbhw9_3/Vendor/vendtmp     0
/tbhw9_3/Vendor/vend2tmp    0
/tbhw9_3/Vendor/changetmp   0
```

Description 3: Here we have 5 inputs : clock, reset, nickel, dime, and quarter, and 3 outputs vend, vend 2 and change. What i did in my code and simulation was showed how inserting the money allows for the state to traverse through the state and eventually vend the item to the user once the right amount of money is inserted. Also if the user inserts more money than is needed then the vending machine will dispense change then reset the amount (or state) to 0. The changes i made were an addition of a second vendable item (vend2) and changed the price of the items (vend1 went from 40 to 35 cents and vend2 is 10 cents).

Videos

https://youtu.be/feMPUwU4MOs

https://youtu.be/yOkVgLIsazE

https://youtu.be/mUX8M905weQ