Faculty of Engineering, Environment and Computing
201CDE/5045CEM Analogue and Digital Electronics 2

Digital electronics coursework

Lecturer

Dr Server Kasap

Student:

Aziz Abdul

ID: 10289484

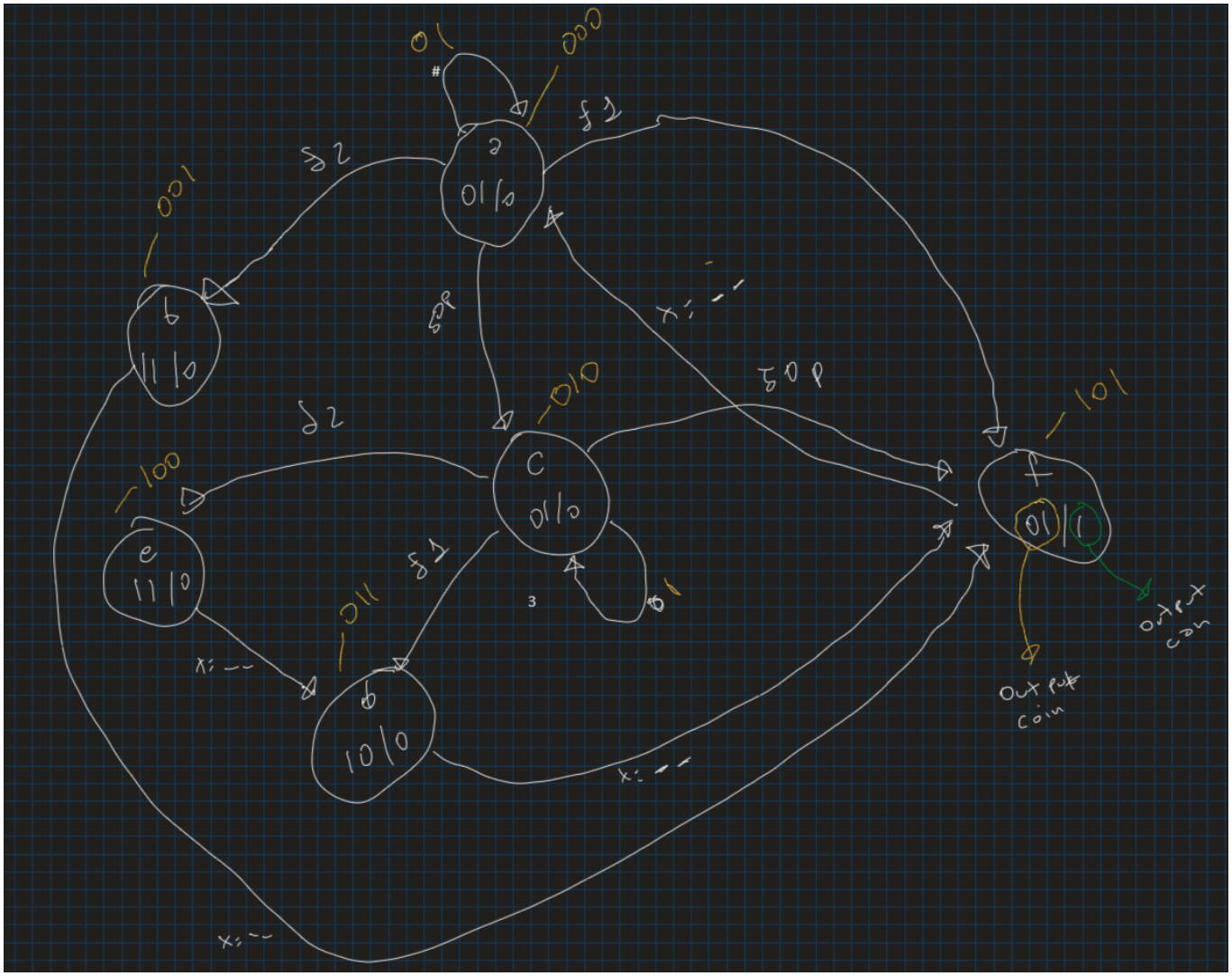
Traditional Design

Last digit of my student ID is 4 therefore this are the values I used:

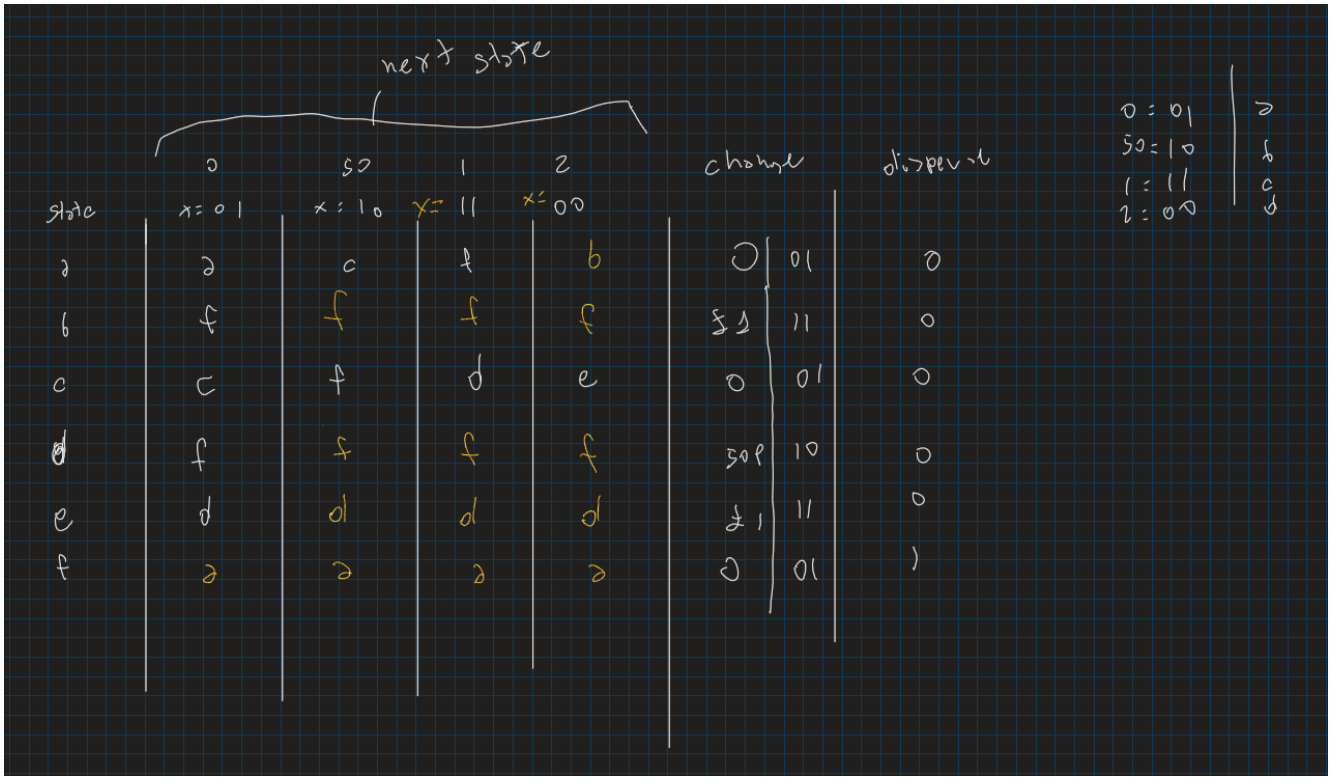
£0 = “01”, 50p = “10”, £1 = ”11”, £2 = “00”

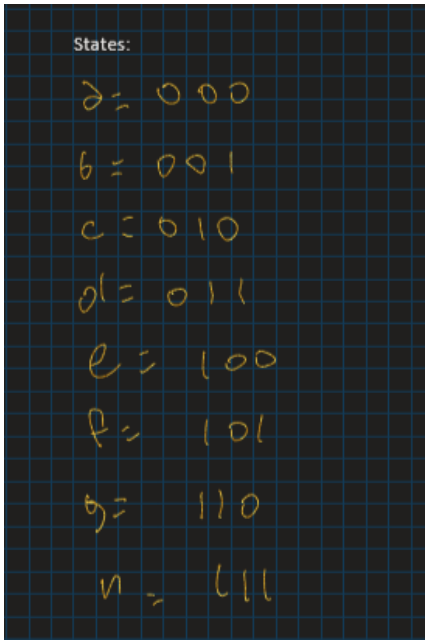
state diagram:

6 States



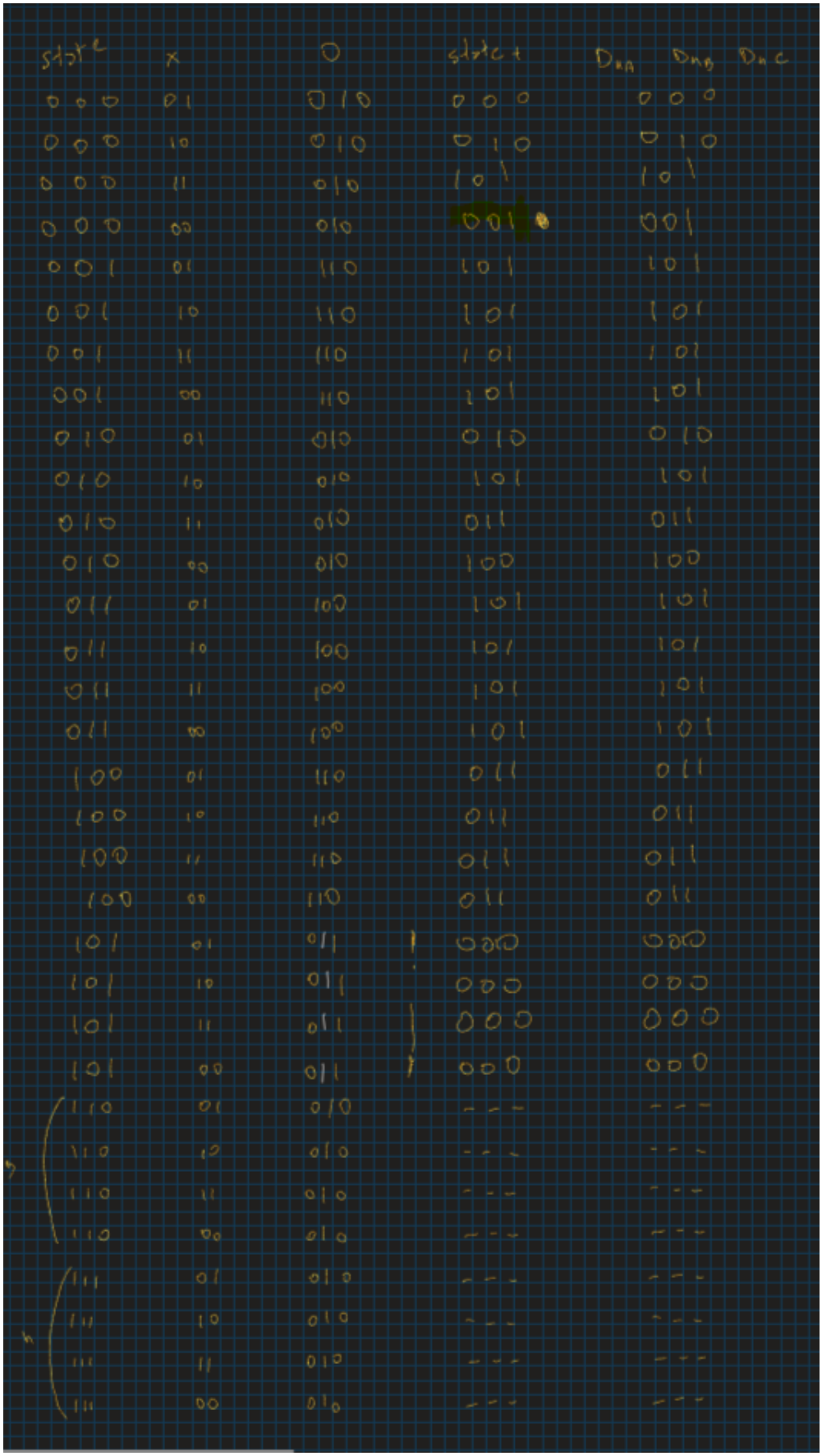
state table:





In total I was able to get an optimum number of 6 states to create the machine. Additional 2 states are created as don't care to fill and create 32 combinational lines in the transitional table

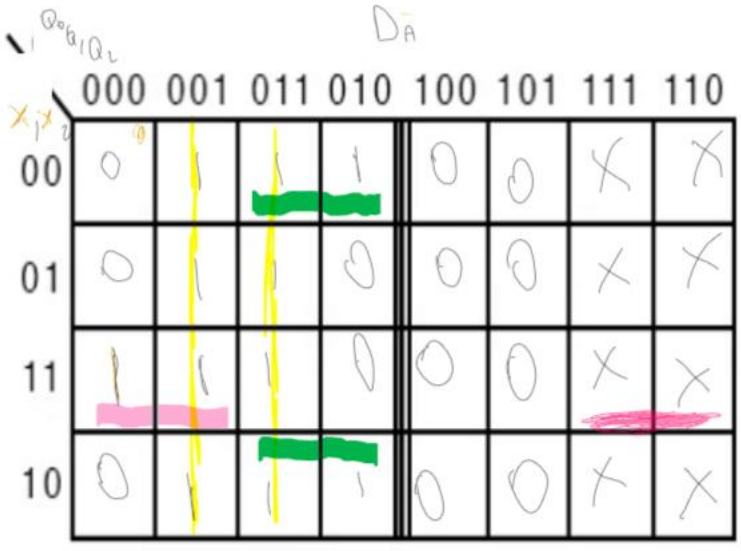
Transitional table



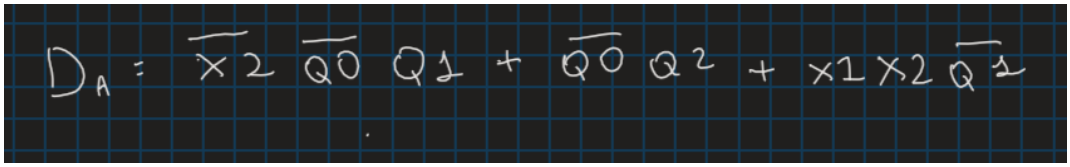
d flip flop excitation table:

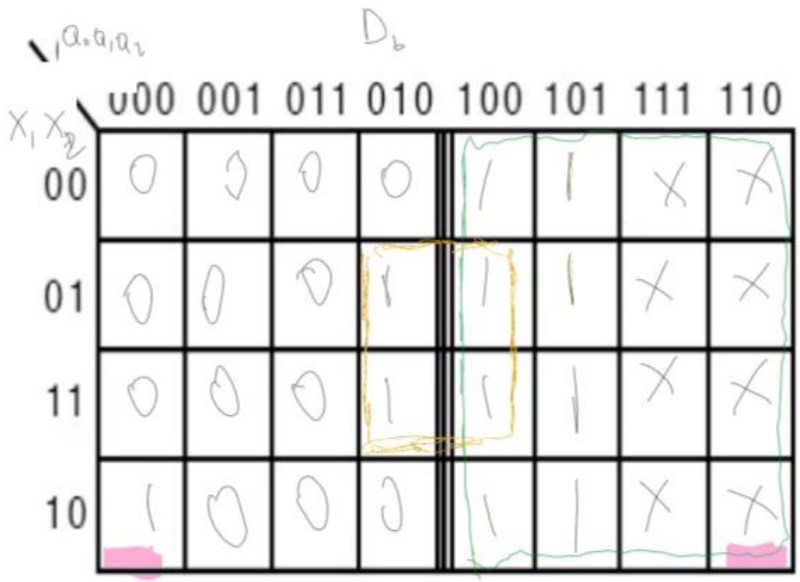
Q Output		Input
$Q_{(n)}$	$Q_{(n+1)}$	D_n
0	0	0
0	1	1
1	0	0
1	1	1

Kmaps:



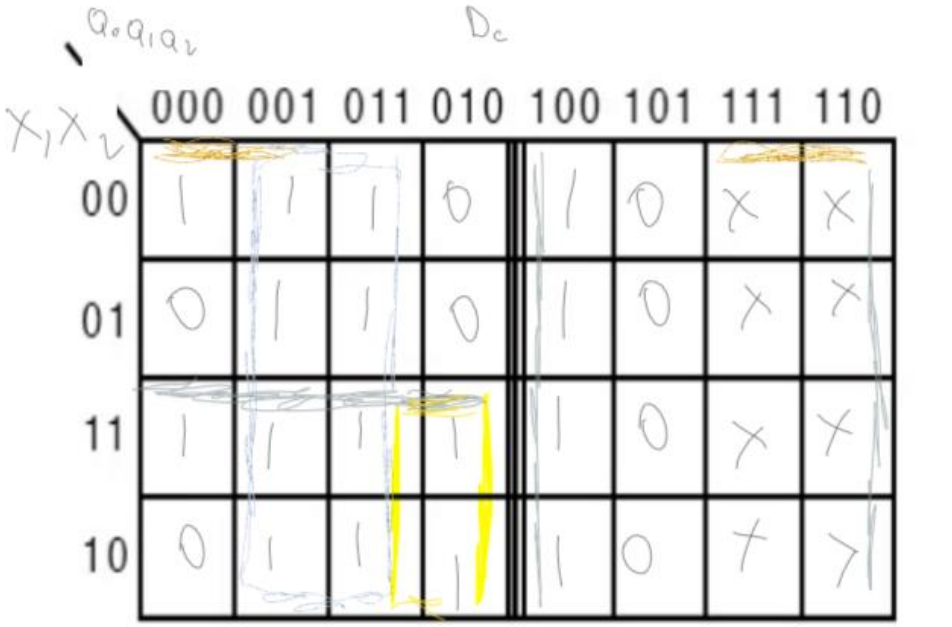
5-variable Karnaugh map (overlay)





5-variable Karnaugh map (overlay)

$$D_B = Q0\,\overline{Q2} + X2\,Q2\,\overline{Q2} + X1\,\overline{X2}\,\overline{Q1}\,\overline{Q2}$$



5-variable Karnaugh map (overlay)

$$D_C = \overline{X1}\,\overline{X2}\,\overline{Q1} + \overline{Q0}\,Q2 + Q0\,\overline{Q2} + X1\,\overline{Q0}\,Q1 + X1X2\,\overline{Q0}$$

output function maps and logic equations:

change (coin out):

O1:

Q_0 \ Q_1Q_2	00	01	11	10
0	0	1	1	0
1	1	0	0	0

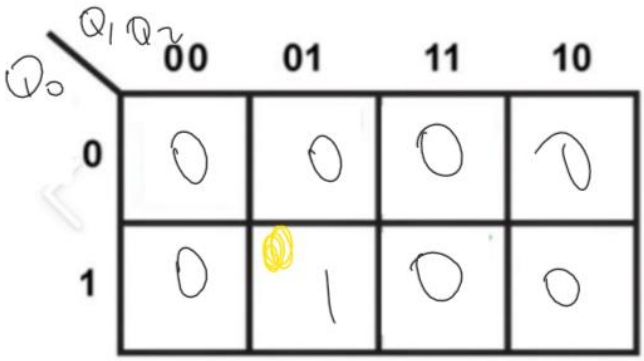
$$O_1 = \overline{Q0}\,Q2 + Q0\,\overline{Q1}\,\overline{Q2}$$

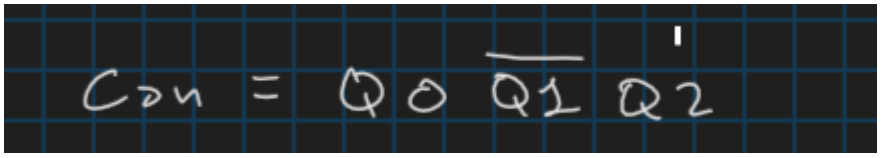
O2:

Q_0 \ Q_1Q_2	00	01	11	10
0	1	1	0	1
1	1	1	1	1

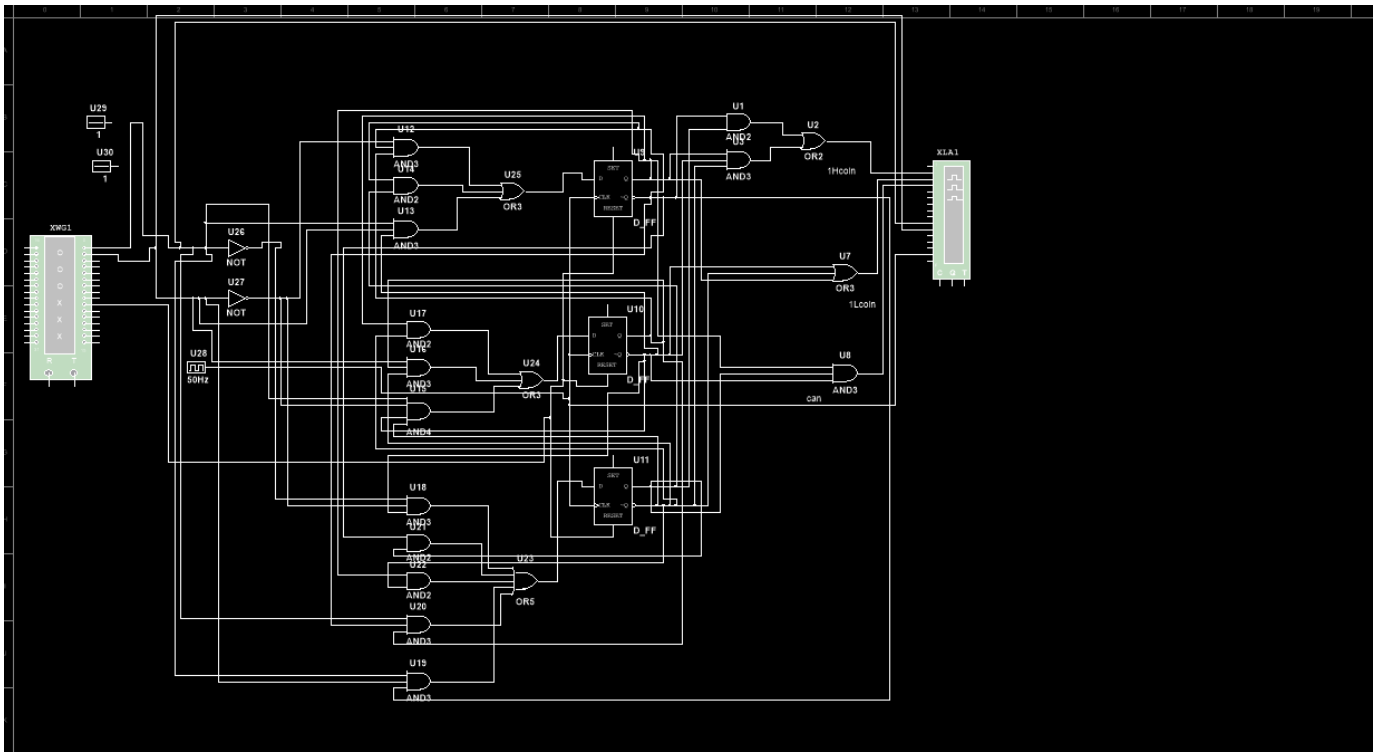
$$Q_2 = \overline{Q0}\,\overline{Q1} + \overline{Q2} + Q0\,Q1$$

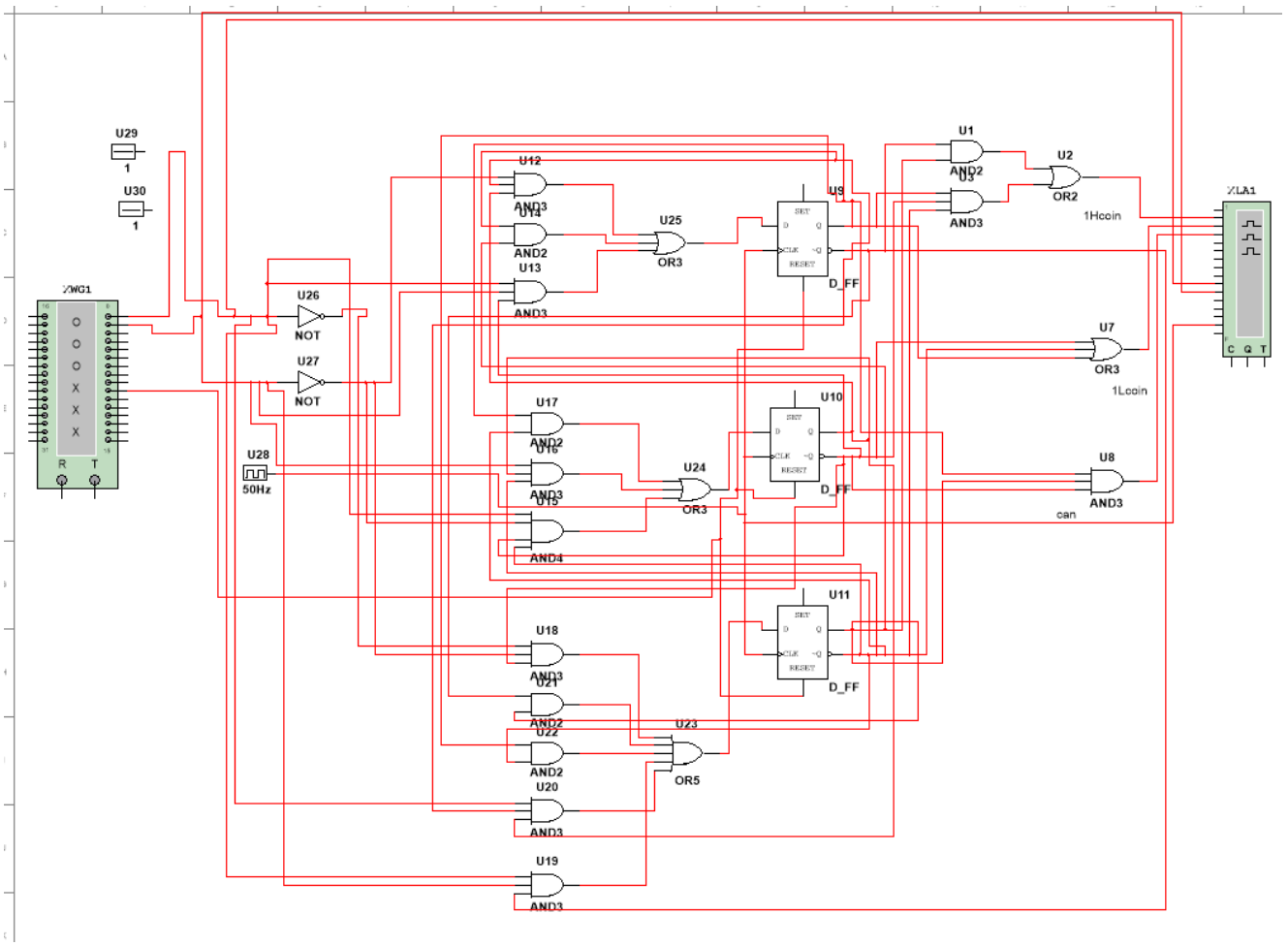
Can out:



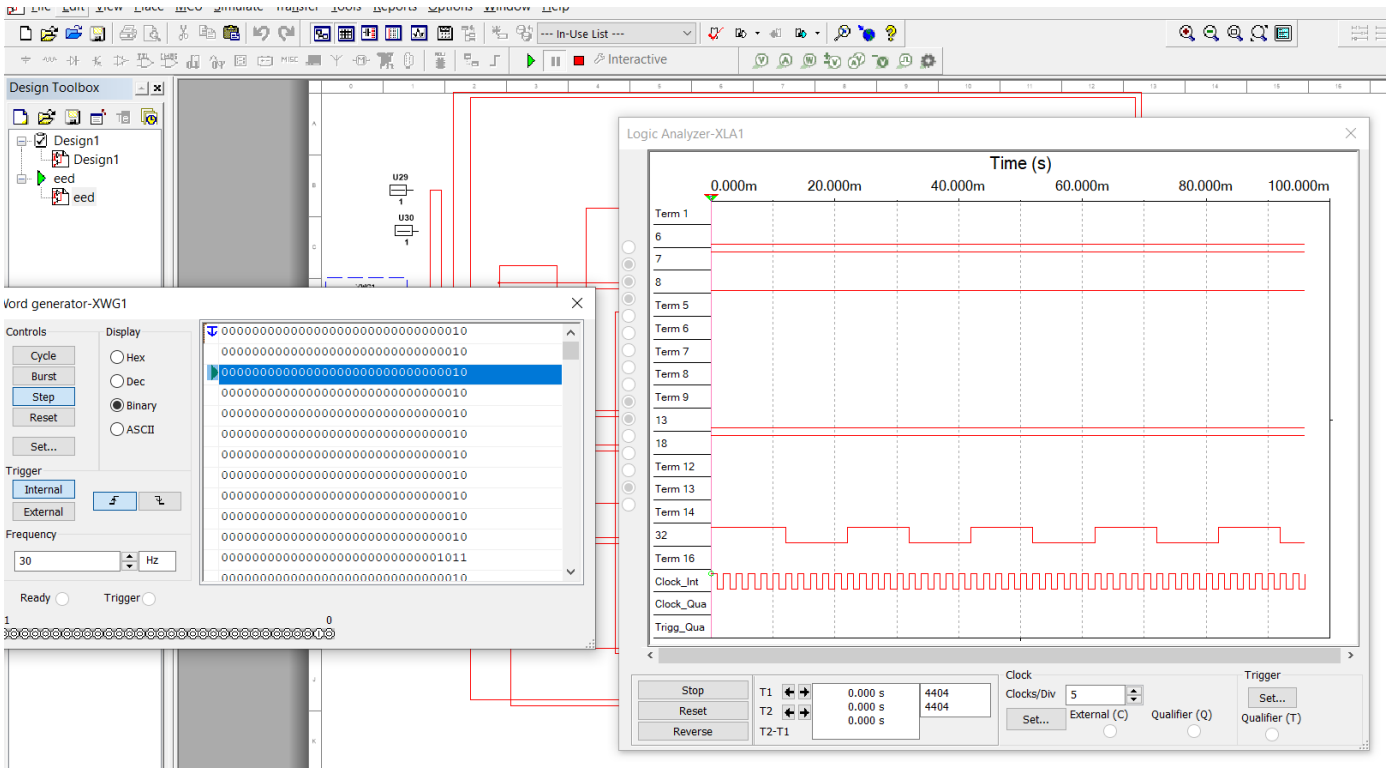


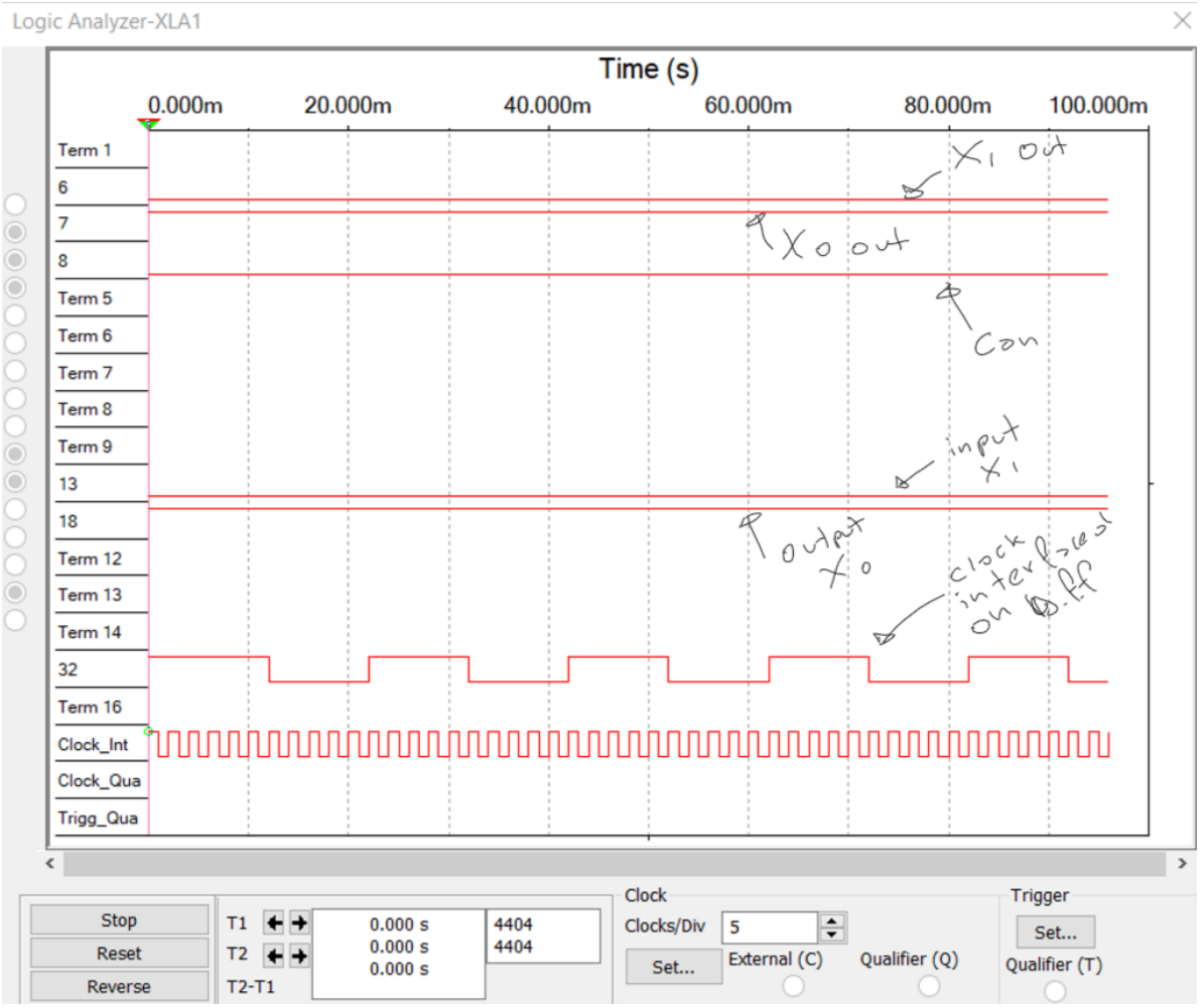
Multisim:



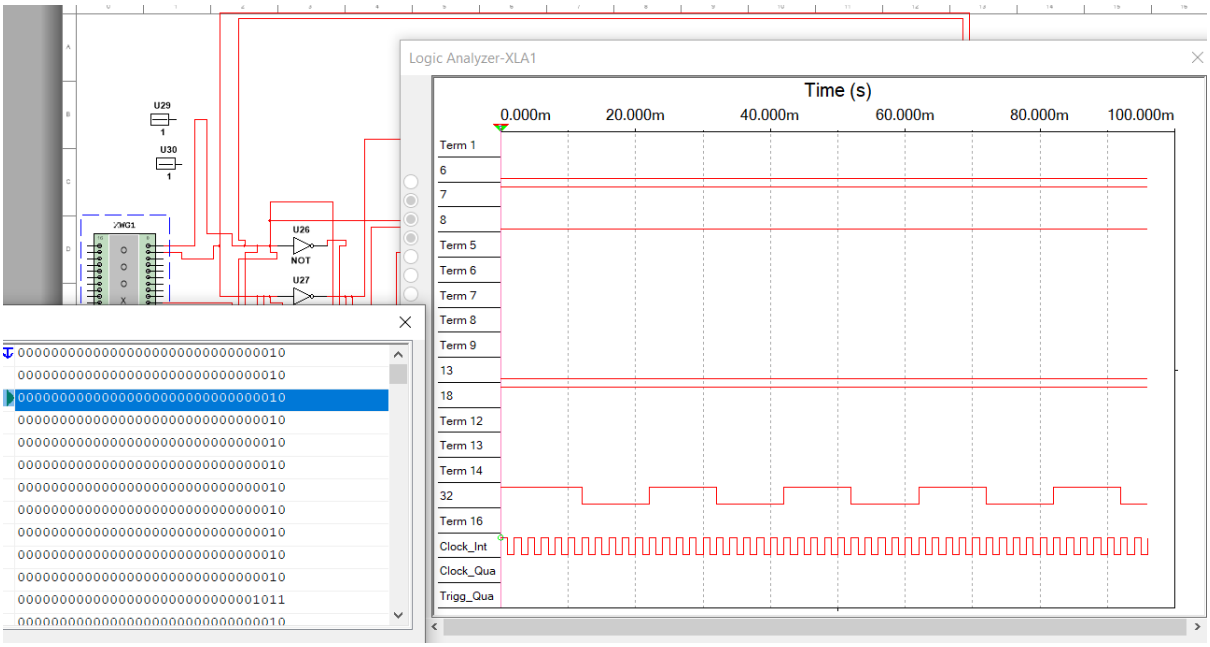


Proof of simulations:





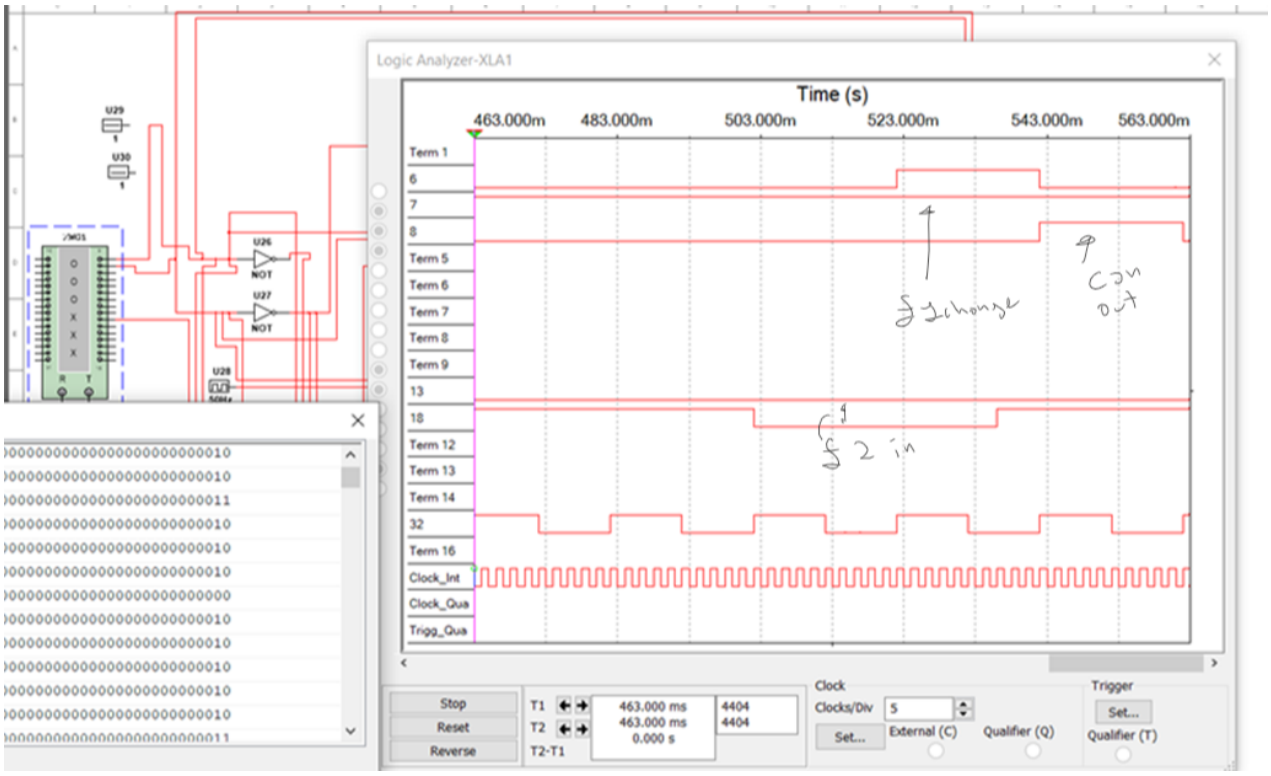
£0 case:



When coin entered is £0, signal 01 (corresponding to £0 for my case) is fed into the logic system. The input X1 = 0 and input X0 = 1 can be seen where, X0

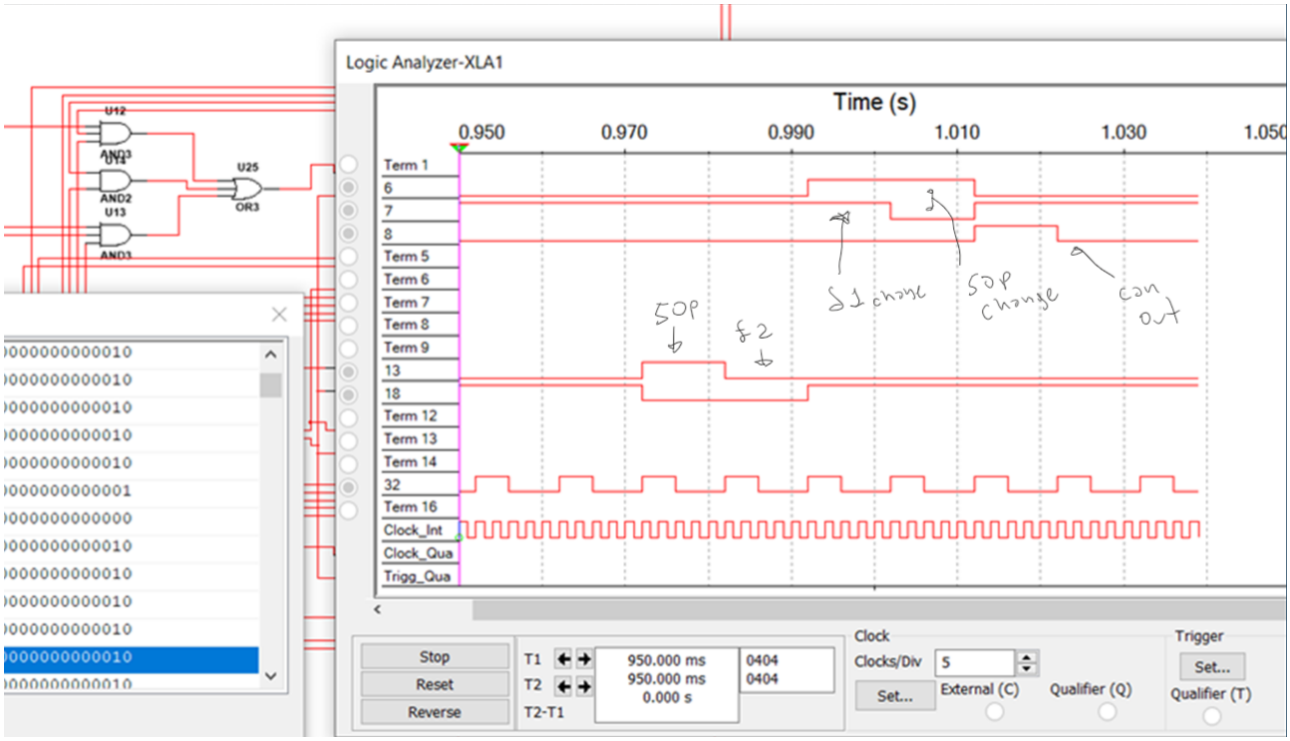
signal is high and X1 signal low. We get no change as it can be seen where output X1X2 = 01 corresponding to £0 change, no can is given (0 state). The output (changes) X1 is low and output X0 is high, representing 01.

£2 case:

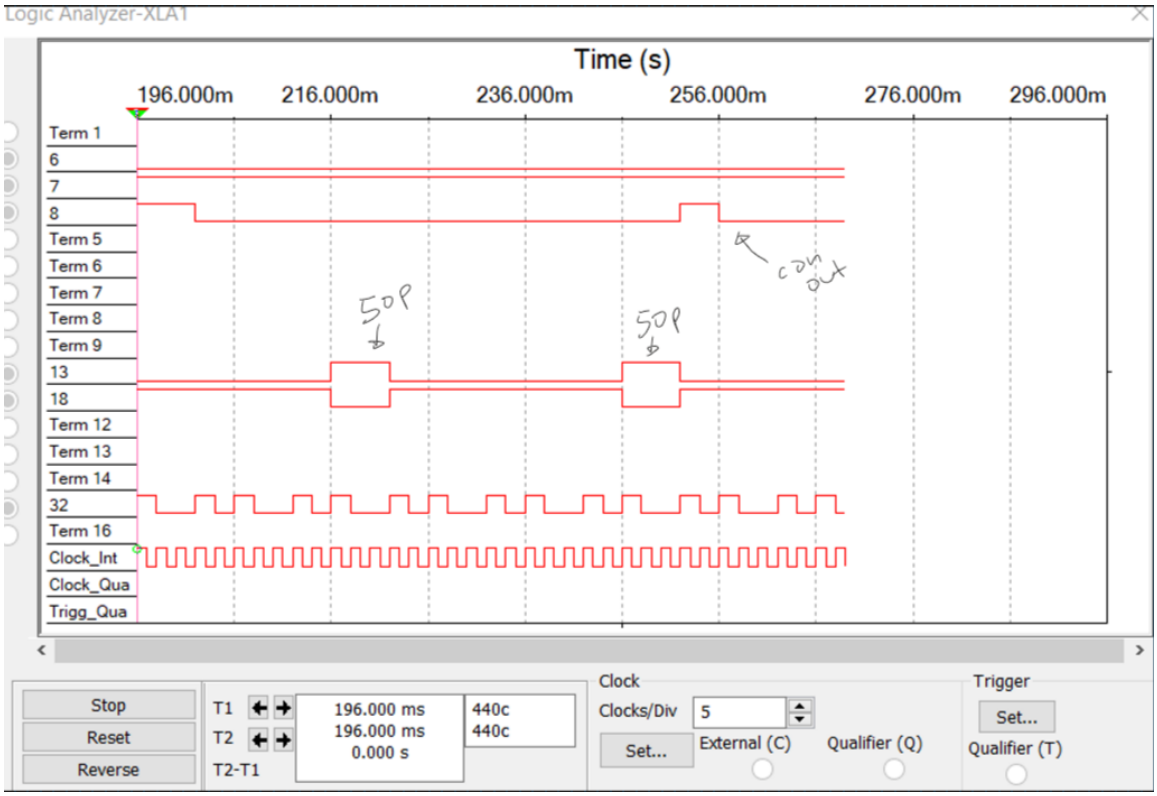


£2 pound is inserted into the machine by taking both inputs waves to 0. The machine then gives £1 change followed by a can.

50p and £2 case:

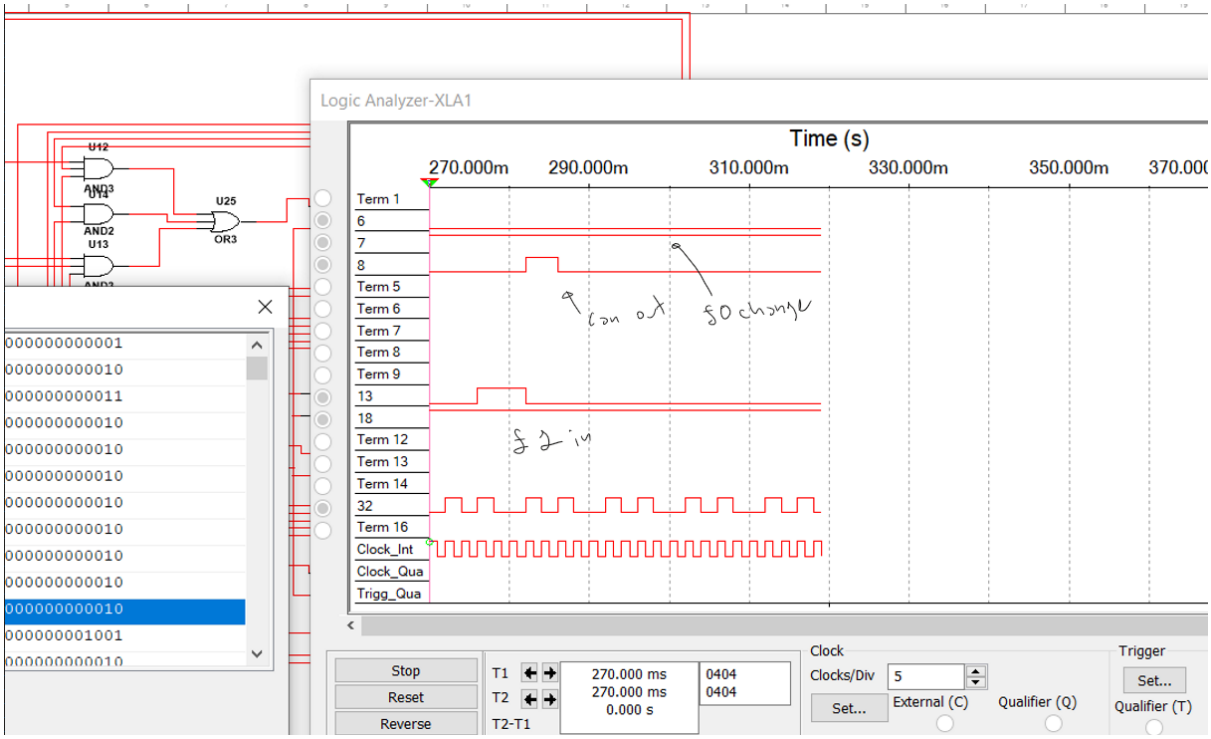


50p and 50p case:

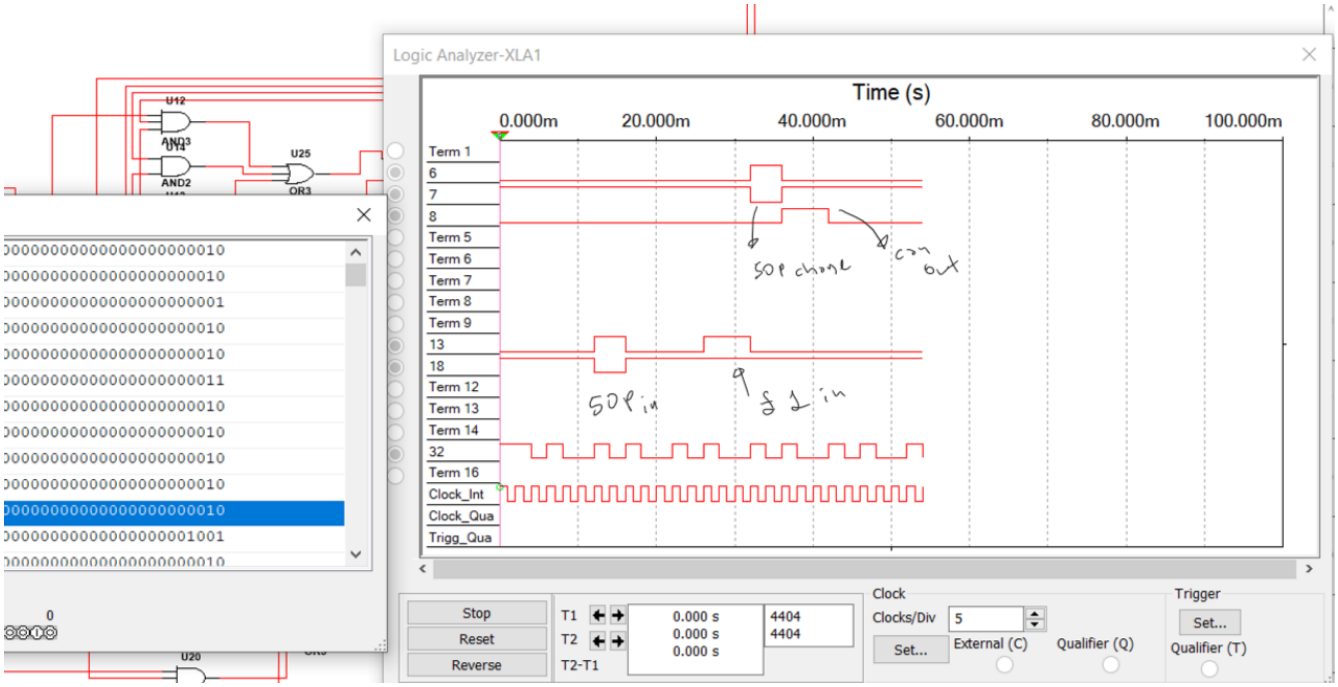


Two 50p inpus add up to £1 therefore a can is dispensed and no change is given.

£1 case:

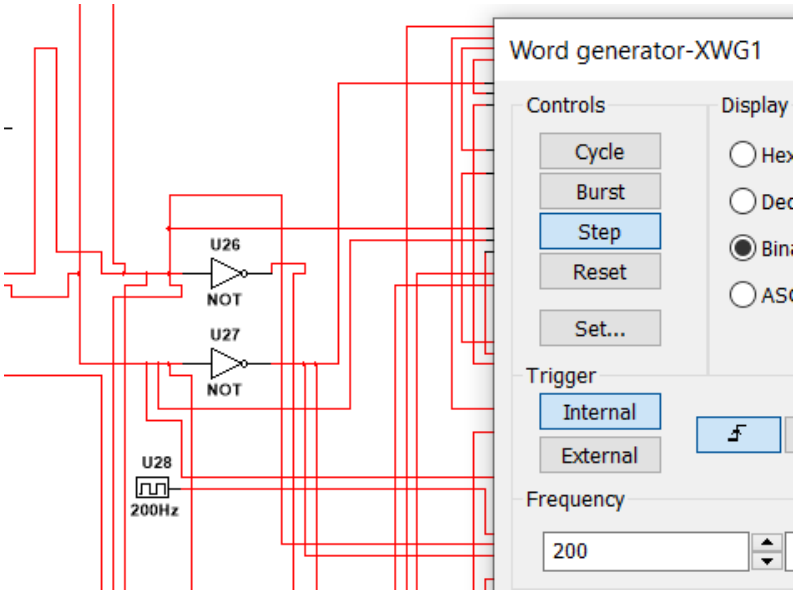


50p and £1 case:



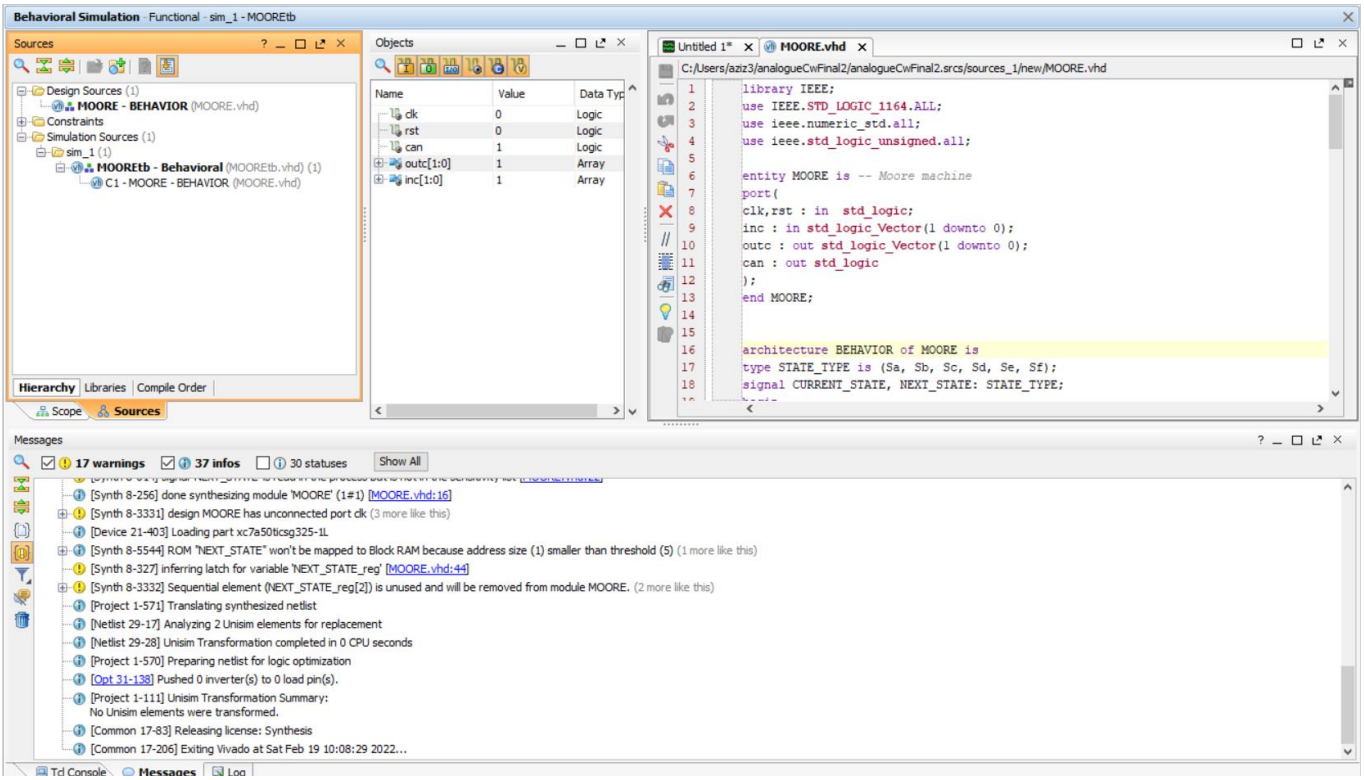
A issues that I encountered was that the memory stored the previous unused input value 50p, therefore when I add the next signal eg £1 pound I get £1.50 input in total therefore I get a wrong waveform. To overcome this issue the

simulation had to be restarted before each test to clean its memory. A simple fix was to add a reset option connected to three D flip flops, A little impulse in the beginning will clear the memory.



The clock frequency connected to D ff is selected to run at 200Hz

VHDL model design with proof of compilation:



VHDL model code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity MOORE is -- Moore machine
port(
clk,rst : in  std_logic;
inc : in std_logic_Vector(1 downto 0);
outc : out std_logic_Vector(1 downto 0);
can : out std_logic
);
end MOORE;

architecture BEHAVIOR of MOORE is
```

```
type STATE_TYPE is (Sa, Sb, Sc, Sd, Se, Sf);
signal CURRENT_STATE, NEXT_STATE: STATE_TYPE;
begin

-- Process to hold synchronous elements (flip-flops)
SYNCH: process(clk)
begin
if(rising_edge(clk)) then
   if(rst = '1') then
   CURRENT_STATE <= Sa;
   else
   CURRENT_STATE <= NEXT_STATE;
   end if;
end if;

CURRENT_STATE <= NEXT_STATE;
end process SYNCH;


-- Process to hold combinational logic
COMBIN: process(CURRENT_STATE, inc)
begin
case CURRENT_STATE is
when Sa =>
outc <= "01";
can <= '0';
if (inc = "01") then -- 0 p
NEXT_STATE <= Sa;
end if;
if (inc = "11") then  --1 pound
NEXT_STATE <= Sf;
end if;
if (inc = "00") then  --2 pound
```

```
NEXT_STATE <= Sb;
end if;
if (inc = "10") then   --50p
NEXT_STATE <= Sc;
end if;

when Sb =>
outc <= "11";
can <= '0';
NEXT_STATE <= Sf;



when Sc =>
outc <= "01";
can <= '0';
if (inc = "01") then  --0p
NEXT_STATE <= Sc;
end if;
if (inc = "11") then --1 pound
NEXT_STATE <= Sd;
end if;
if (inc = "00") then --2 pound
NEXT_STATE <= Se;
end if;
if (inc = "10") then
NEXT_STATE <= Sf;
end if;

when Se =>
outc <= "11";
can <= '0';
NEXT_STATE <= Sd;
```

```
when Sd =>
outc <= "10";

can <= '0';
NEXT_STATE <= Sf;

when Sf =>
outc <= "01";

can <= '1';
NEXT_STATE <= Sa;

end case;
end process COMBIN;

end BEHAVIOR;
```

---

end of code

VHDL test bench:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity MOOREtb is
--  Port ( );
end MOOREtb;

architecture Behavioral of MOOREtb is
component MOORE is -- Moore machine
port(
clk, rst : in  std_logic;
inc : in std_logic_Vector(1 downto 0);
outc : out std_logic_Vector(1 downto 0);
can : out std_logic
);
end component;

signal clk, rst, can :  std_logic := '0';
signal outc : std_logic_Vector(1 downto 0) := "00";
signal inc : std_logic_Vector(1 downto 0) := "00";



begin
C1 : MOORE port map (clk => clk, inc => inc, outc => outc, rst => rst, can => can);

clk_process:process(clk)
begin
clk <= not clk after 10 ns;   --oscillate at this specific period
```

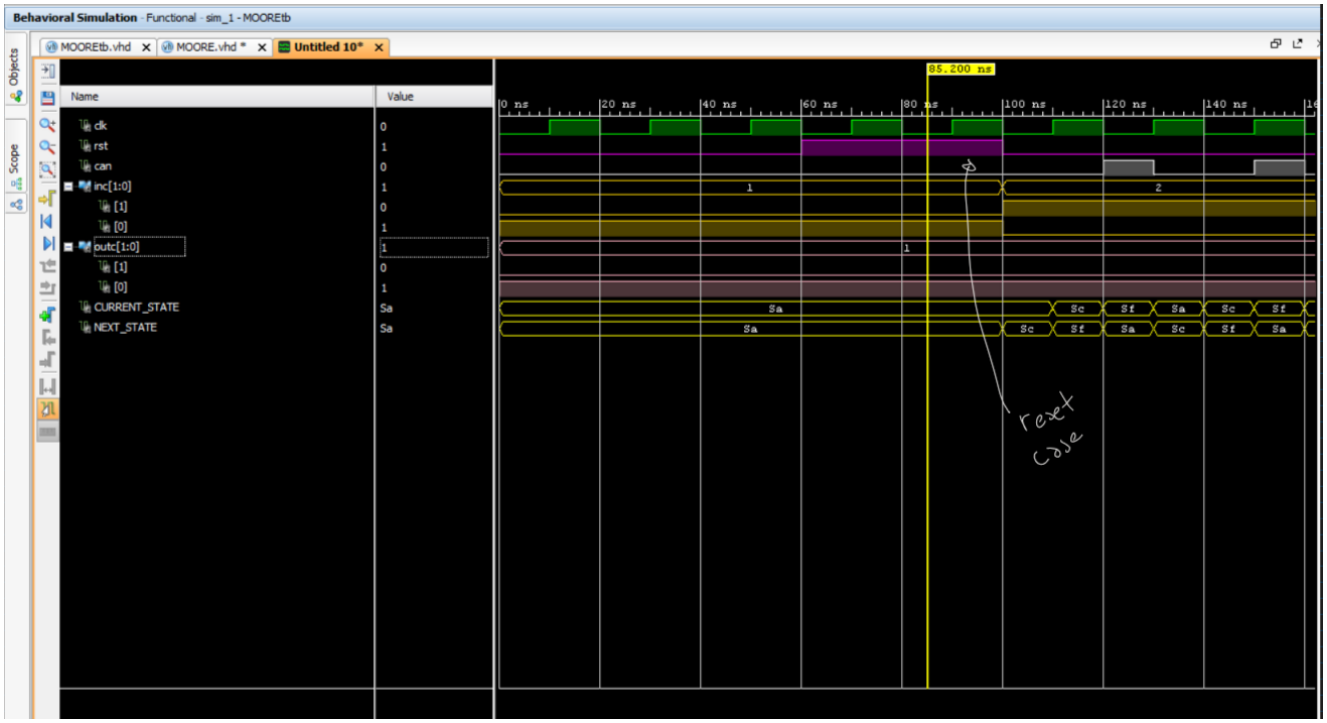
```
end process clk_process;

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

control_process:process

begin



inc <= "01";

wait for 50 ns;

rst <= '1' after 10 ns ;

wait for 50 ns;

rst <= '0';

inc <= "10";

wait for 50 ns;



--£50p case

inc <= "10";

wait for 50 ns;

inc <= "01";

wait for 50 ns;



--- £50p

inc <= "10";

wait for 50 ns;
```

```
-----£1


inc <= "11";
wait for 50 ns;


-----£2 and 50p
inc <= "10";
wait for 50 ns;
inc <= "00";
wait for 50 ns;


-----£1.50
inc <= "10";
inc <= "11";
wait for 50 ns;
end process control_process;




end Behavioral;
```

End of test bench code

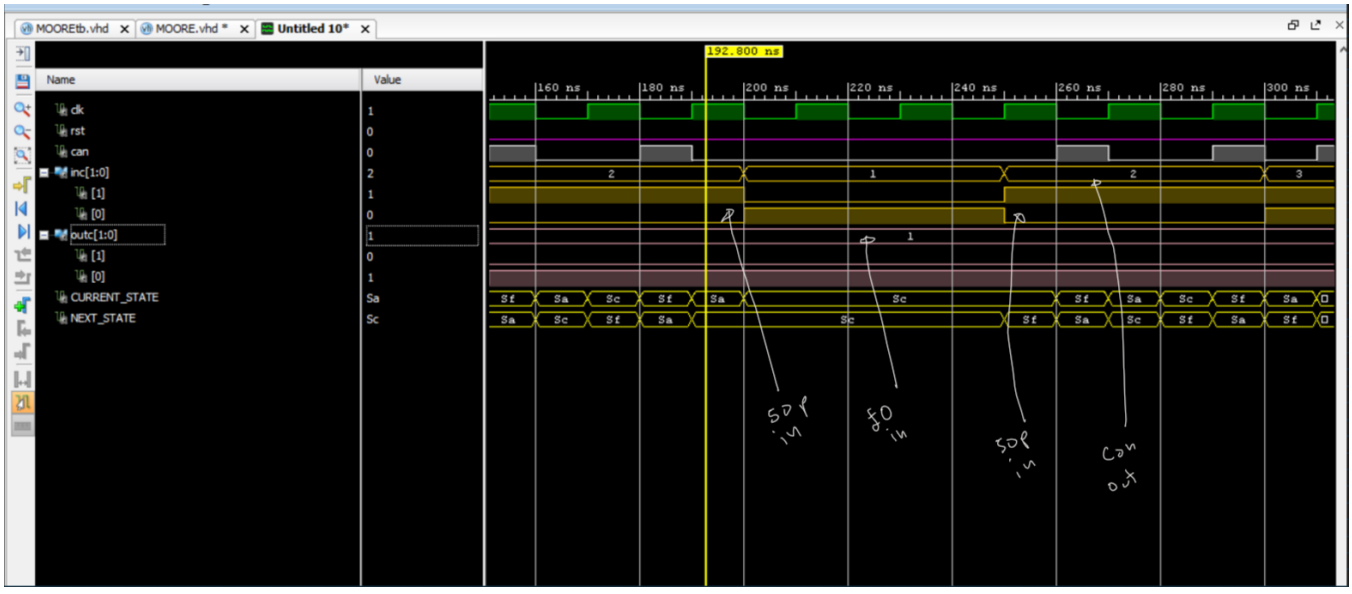
Simulation:

Reset case:



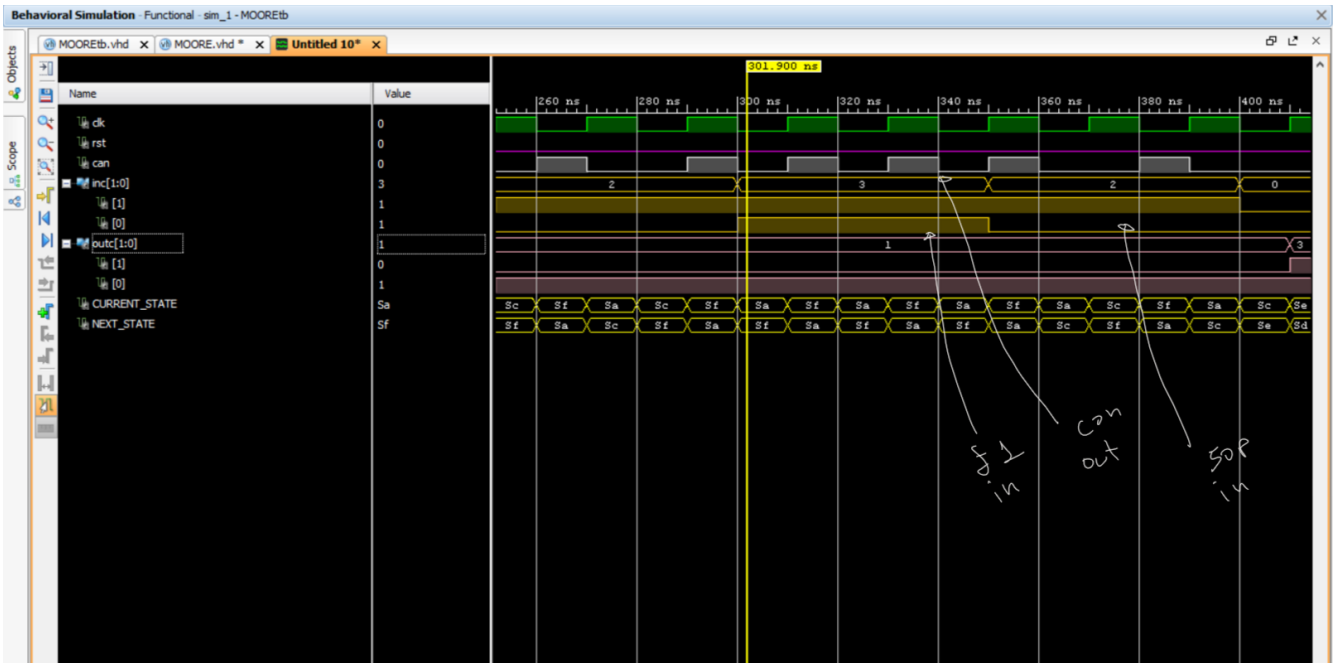
A small reset impulse is sent by the test bench as it can be seen above to fulfil the task requirement.

50p then £0 and later 50p in:



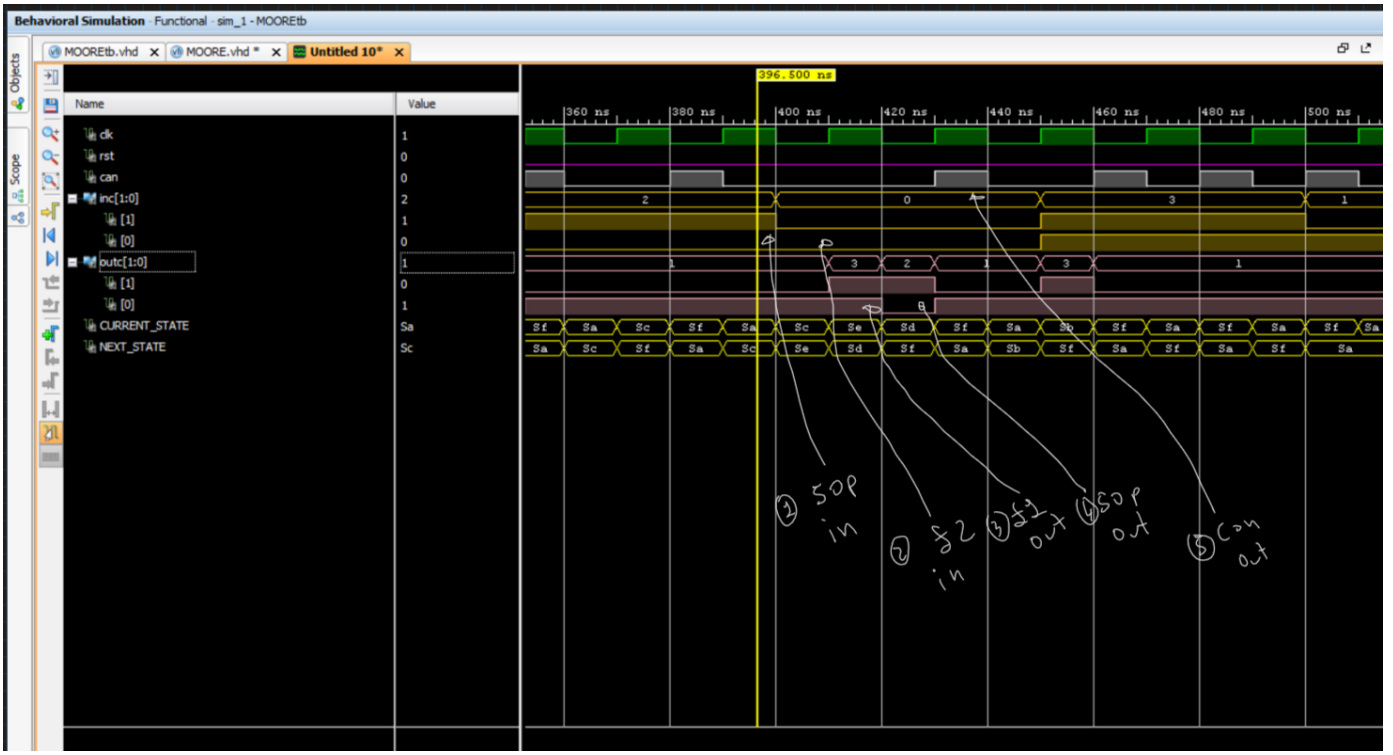
When the system has received the state changes from Sa to Sc and wait in there while £0 has been inserted, later when 50p has been inserted the the state changes from Sc to Sf and it returns to the initial state Sa. We get no change. This waveform confirms the correct state jumps.

Just £1 pound in and later 50p in case:



When £1 pound is inserted, the machine goes from Sa directly to Sf. This where a can is dispensed and it returns to the initial state Sa and later goes to Sc as 50p is inserted.

50p in then £2 in case:



Just 50p case:

50p and £1 case: