

LOGIC PROJECT

Vending Machine

DR. Howida Abd Ailatif

ASSIGNMENT Title:

Vending Machine

No.	Name & Email	Academic Number	Task Completed
1.2	Mahmoud Galal Ramadan El-Gendy	20812021101328	Code, Simulation, Schematic diagram using Active HDL
	mahmoudgalal173.95@gmail.com		
2.1	Hassan Mohamed Hassan Ali	20812021101068	
	hassanalzyat.contact@gmail.com		
3.2	Karim Salah Eldin Aboumesalam	20812021101453	
	kimos20139@gmail.com		
4.2	Menna Allah Essam Ahmed Salem	20812021200937	20TK (Truth Tables - K-Maps Boolean Functions)
	me762927@gmail.com		
5.2	Mohamed Gehad Hussien Metwally	20812021100025	Document, State Diagram
	engmgehad@gmail.com		
6.2	Nancy Ayman Nabil Mohamed	20812021200506	10TK – 15TK (Truth Tables K-Maps Boolean Functions)
	na8594761@gmail.com		
7.1	Reham Hamdy Mohamed Ibrahim	20812021201109	
	rehamhamdi53@gmail.com		
8.1	Shahd AlSayed Ahmed Ali	20812021200543	
	shahdelsayed3303@gmail.com		
9.1	Shahd AbdelNabi Mahmoud AbdelNabi	20812021200898	Design using Proteus
	shahdelshamy964@gmail.com		
10.1	Shehab Shukri AbdelNabi Ibrahim	20812021101006	
	sh.s.ghazal2003@gmail.com		

“The assignment Aim”

A vending machine is a self-service device that allows customers to purchase products or services without the need for human intervention. It is a common sight in various public spaces such as schools, offices, airports, train stations, and shopping centers.

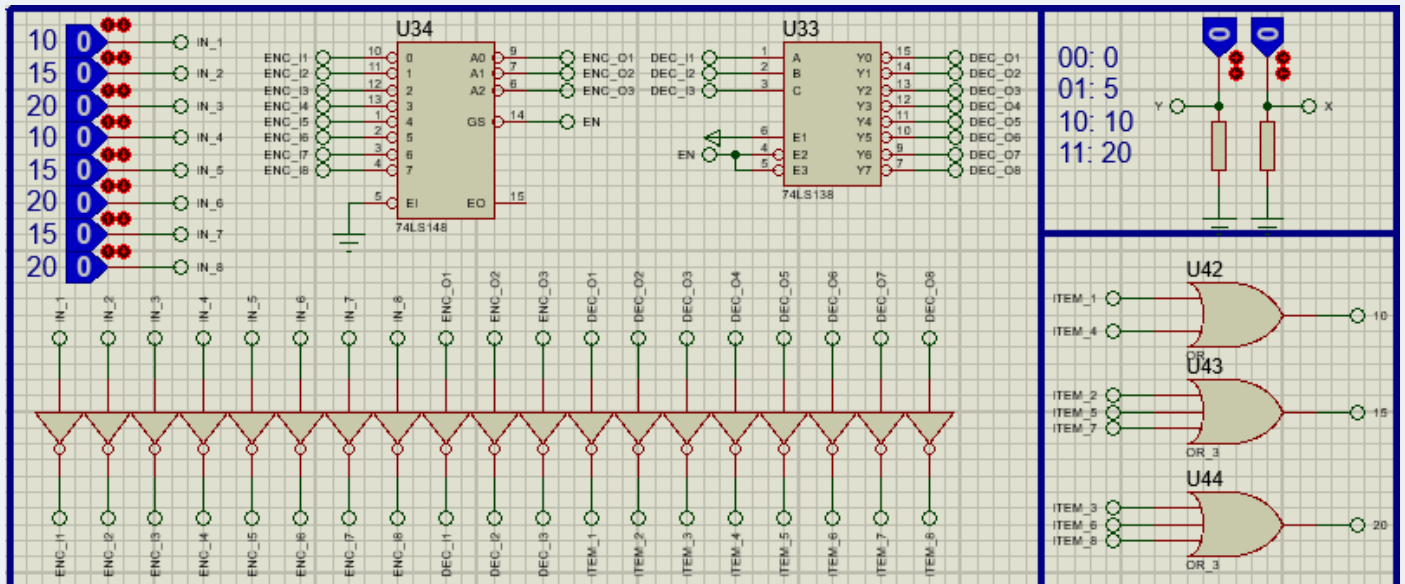
The primary purpose of a vending machine is to provide convenience and accessibility to customers. It typically consists of a cabinet or enclosure that houses the products, a selection mechanism, a payment system, and a dispensing mechanism.

The content of our vending machine:

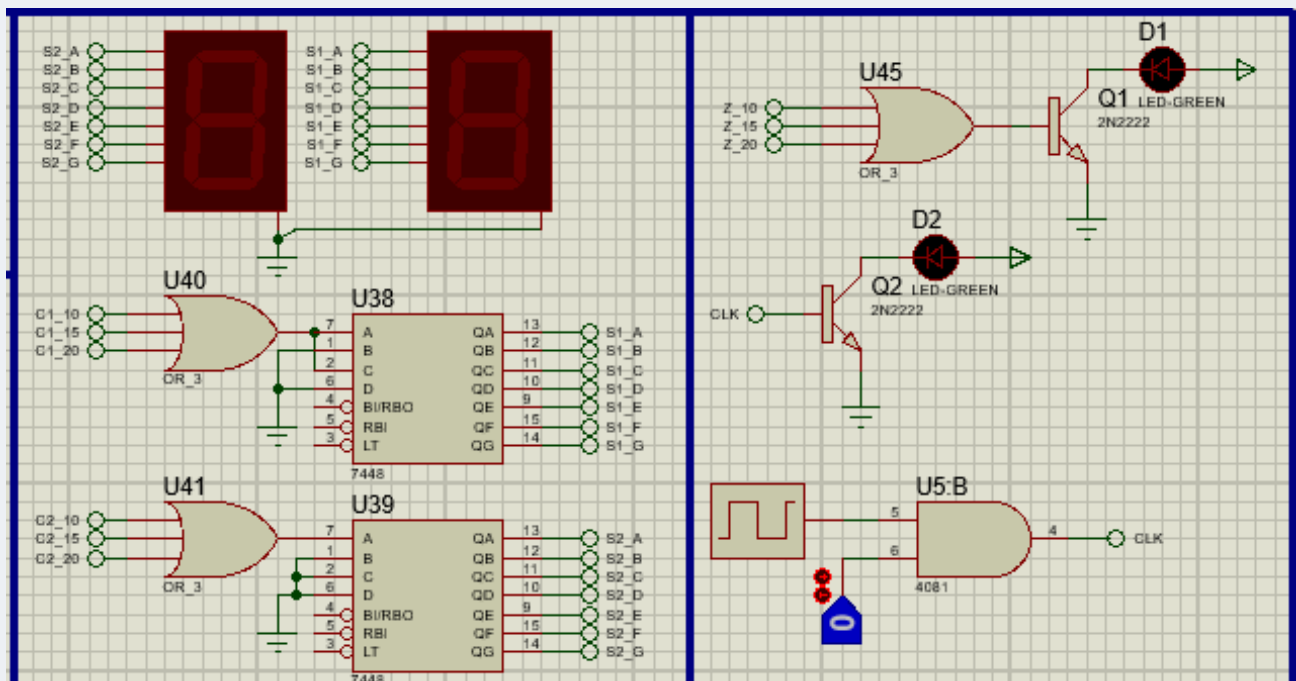
contains 3 products. You can enter prices of 10, 15, and 20 If you enter price that bigger than the product price you will get the product and the exchange of your many. If you enter price less than product price the Machine will wait to the rest for specific time if you don't the process will be canceled, and machine will get out your money.

The Problem Solution

Inputs:



Outputs:



For 10TK

Truth Tables:

Q	Y	X	Q*	Z	C2	C1
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	1	1	0	1	1	0
1	0	0	0	0	0	1
1	0	1	0	1	0	0
1	1	0	0	1	0	1
1	1	1	0	1	1	1

State Diagram:



K-Maps & Boolean Functions:

Q \ YX	00	01	11	10
0	0	1	0	0
1	0	0	0	0

$$Q^* = Q'Y'X$$

Q \ YX	00	01	11	10
0	0	0	1	1
1	0	1	1	1

$$Z = Y + QX$$

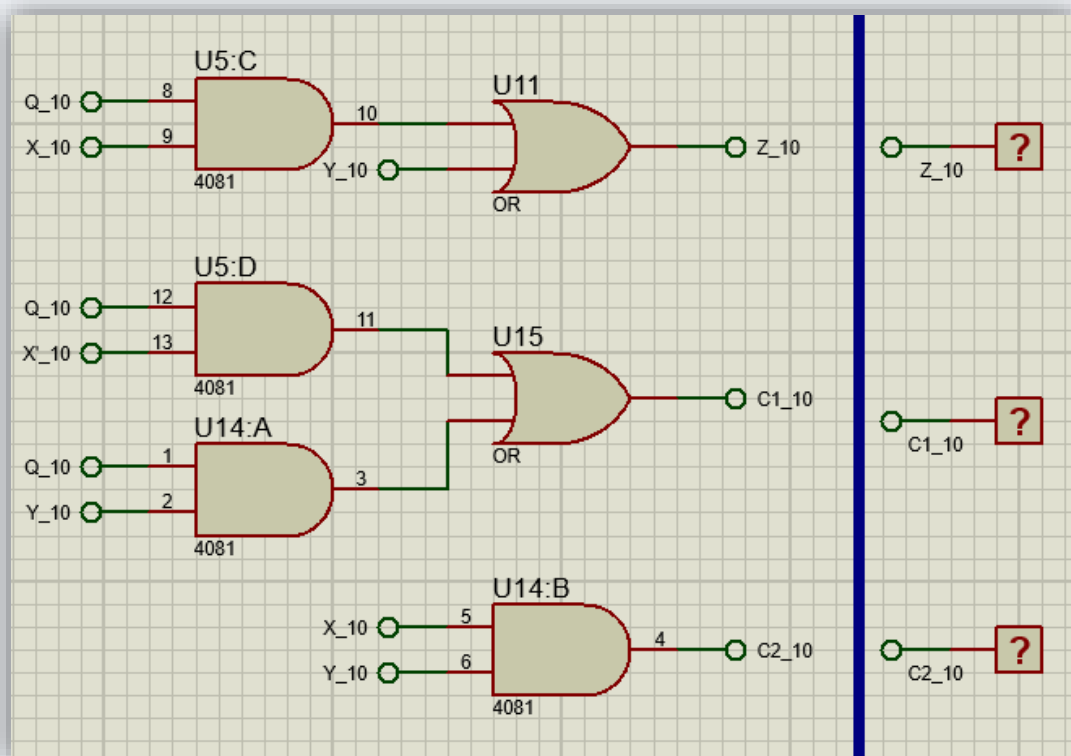
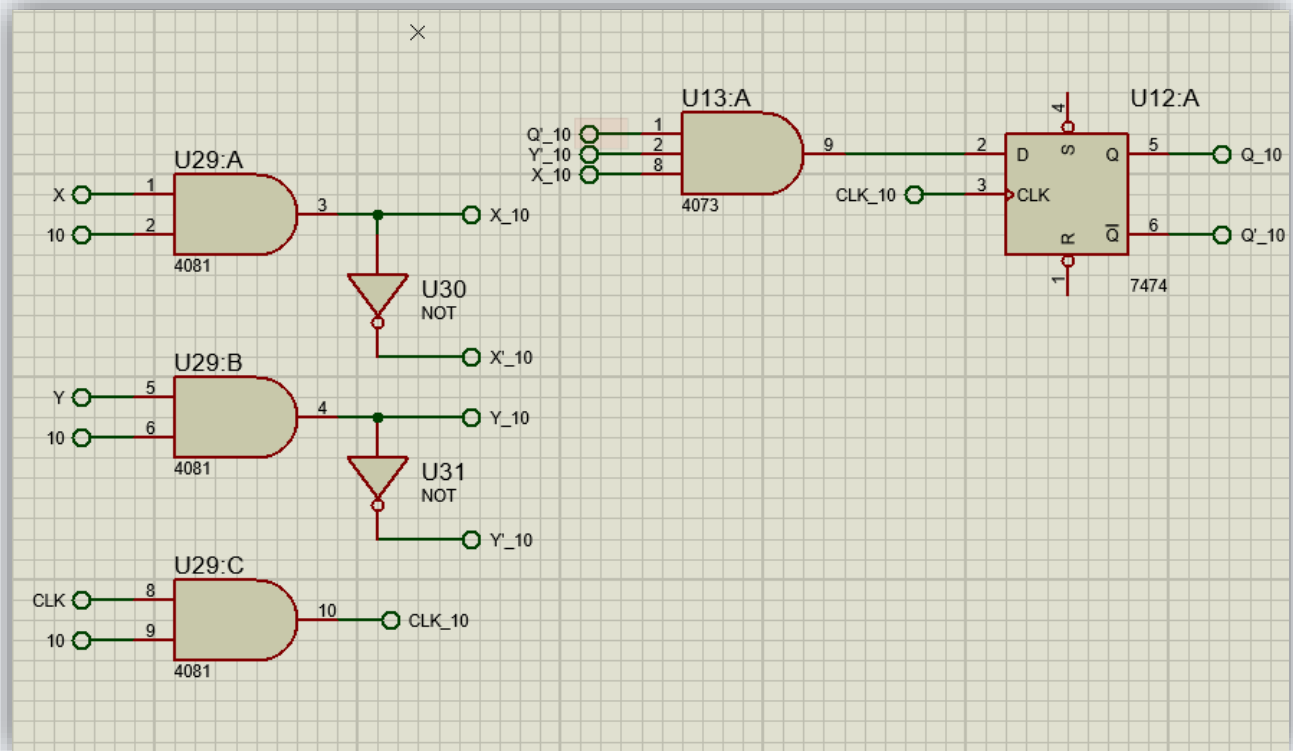
Q \ YX	00	01	11	10
0	0	0	1	0
1	0	0	1	0

$$C2 = XY$$

Q \ YX	00	01	11	10
0	0	0	0	0
1	1	0	1	1

$$C1 = QY + QX'$$

Logic Circuits Design:

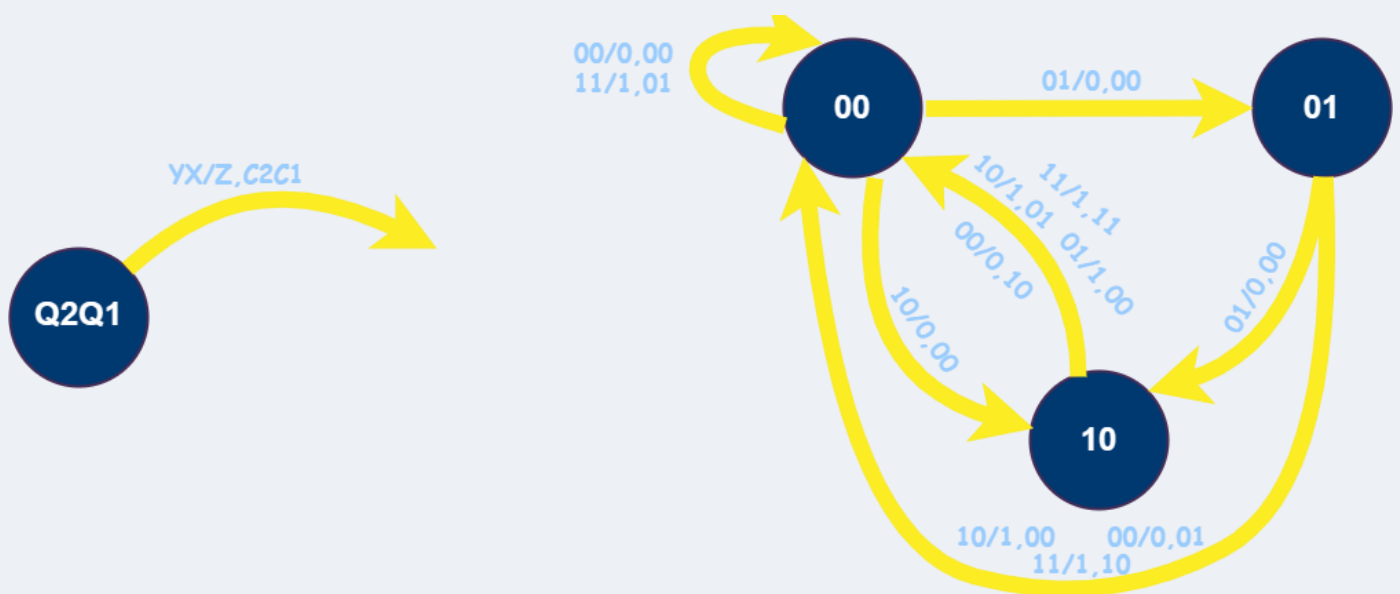


For 15TK

Truth Tables:

Q2	Q1	Y	X	Q2*	Q1*	Z	C2	C1
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	1	0	0	0	0
0	0	1	1	0	0	1	0	1
0	1	0	0	0	0	0	0	1
0	1	0	1	1	0	0	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	1	1	0
1	0	0	0	0	0	0	1	0
1	0	0	1	0	0	1	0	0
1	0	1	0	0	0	1	0	1
1	0	1	1	0	0	1	1	1
1	1	0	0	X	X	X	X	X
1	1	0	1	X	X	X	X	X
1	1	1	0	X	X	X	X	X
1	1	1	1	X	X	X	X	X

State Diagram:



K-Maps & Boolean Functions:

YX Q ₂ Q ₁	00	01	11	10
00	0	0	0	1
01	0	1	0	0
11	X	X	X	X
10	0	0	0	0

$$Q_2^* = Q_2'Q_1'YX' + Q_1Y'X$$

YX Q ₂ Q ₁	00	01	11	10
00	0	1	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	0	0

$$Q_1^* = Q_2'Q_1'Y'X$$

YX Q ₂ Q ₁	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	X	X	X	X
10	0	1	1	1

$$Z = YX + Q_1Y + Q_2X + Q_2Y$$

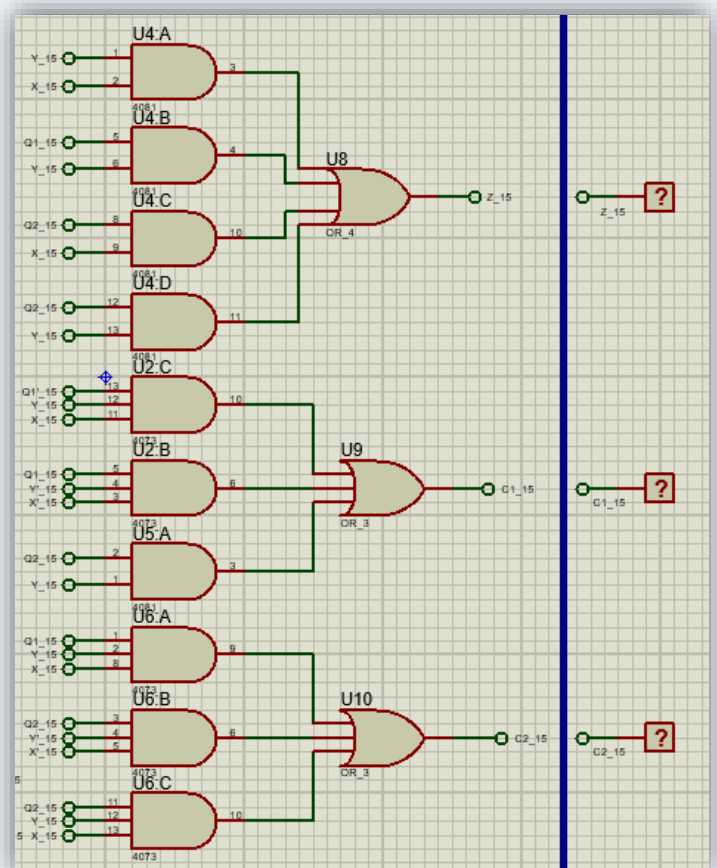
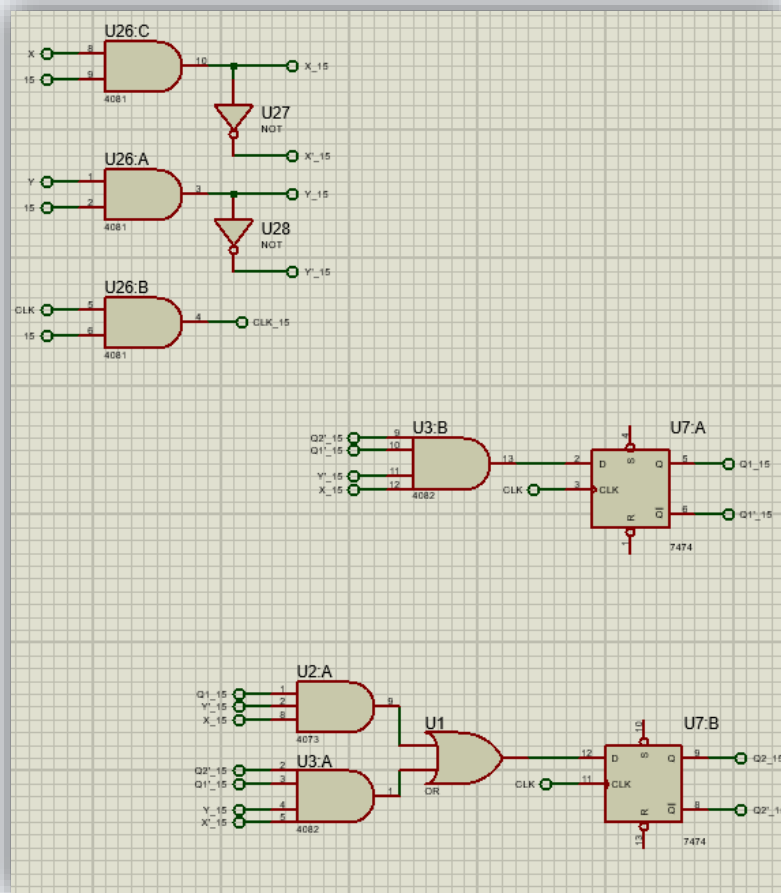
YX Q ₂ Q ₁	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	X	X	X	X
10	1	0	1	0

$$C_2 = Q_2Y'X' + Q_1YX + Q_2YX$$

YX Q ₂ Q ₁	00	01	11	10
00	0	0	1	0
01	1	0	0	0
11	X	X	X	X
10	0	0	1	1

$$C_1 = Q_2Y + Q_1'YX + Q_1Y'X'$$

Logic Circuits Design:

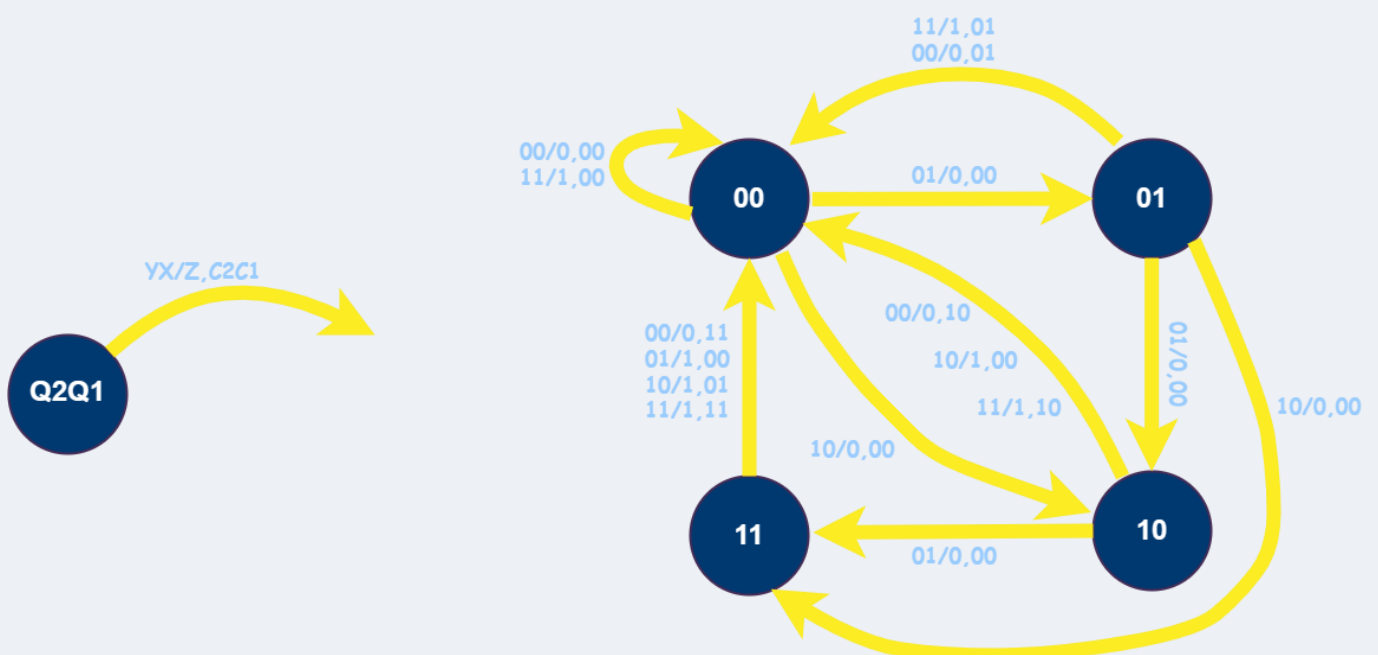


For 20TK

Truth Tables:

Q2	Q1	Y	X	Q2*	Q1*	Z	C2	C1
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	1	0	0	0	0
0	0	1	1	0	0	1	0	0
0	1	0	0	0	0	0	0	1
0	1	0	1	1	0	0	0	0
0	1	1	0	1	1	0	0	0
0	1	1	1	0	0	1	0	1
1	0	0	0	0	0	0	1	0
1	0	0	1	1	1	0	0	0
1	0	1	0	0	0	1	0	0
1	0	1	1	0	0	1	1	0
1	1	0	0	0	0	0	1	1
1	1	0	1	0	0	1	0	0
1	1	1	0	0	0	1	0	1
1	1	1	1	0	0	1	1	1

State Diagram:



K-Maps & Boolean Functions:

YX Q ₂ \ Q ₁	00	01	11	10
00	0	0	0	1
01	0	1	0	1
11	0	0	0	0
10	0	1	0	0

$$Q_2^* = Q_2'YX' + Q_2'Q_1Y'X + Q_2Q_1'Y'X$$

YX Q ₂ \ Q ₁	00	01	11	10
00	0	1	0	0
01	0	0	0	1
11	0	0	0	0
10	0	1	0	0

$$Q_1^* = Q_1'Y'X + Q_2'Q_1YX'$$

YX Q ₂ \ Q ₁	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	1

$$Z = YX + Q_2Y + Q_2Q_1X$$

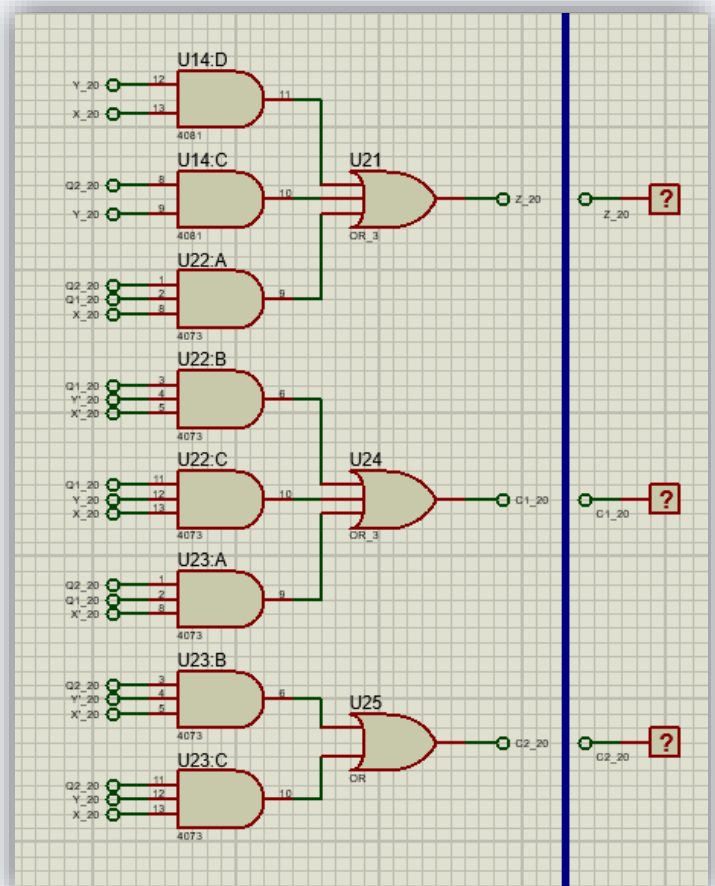
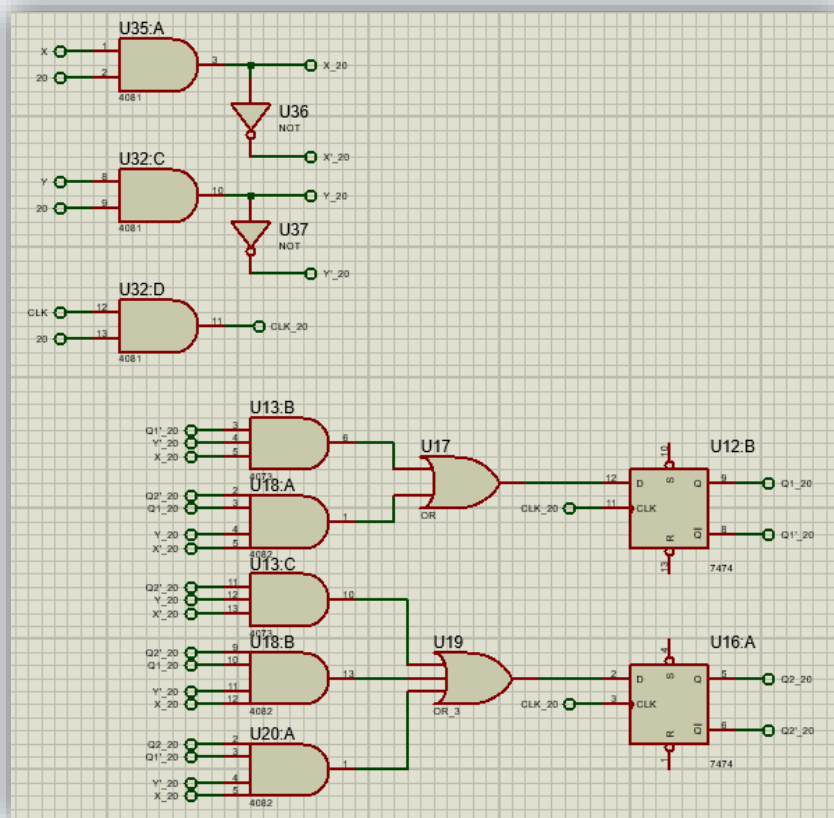
YX Q ₂ \ Q ₁	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	0	1	0
10	1	0	1	0

$$C_2 = Q_2Y'X' + Q_2YX$$

YX Q ₂ \ Q ₁	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	1	0	1	1
10	0	0	0	0

$$C_1 = Q_2Q_1X' + Q_1YX + Q_1Y'X'$$

Logic Circuits Design:



Implementation

Implementation the system using Verilog HDL:

20:

```
1  module VendingMachine_20 (clock, reset, price_1, price_2, out, change_1, change_2);
2      input clock;
3      input reset;
4      input price_1, price_2;
5
6      output reg out;
7      output reg change_1, change_2;
8
9      parameter A = 2'b00;
10     parameter B = 2'b01;
11     parameter C = 2'b10;
12     parameter D = 2'b11;
13
14     reg [1:0] currentState;
15     reg [1:0] nextState;
16
17     always @(posedge clock)
18     begin
19         if (reset)
20         begin
21             currentState <= A;
22             nextState <= A;
23             out <= 1'b0;
24             change_1 <= 1'b0;
25             change_2 <= 1'b0;
26         end
27     else
28     begin
29         currentState <= nextState;
30         case(currentState)
31             A: case({price_2, price_1})
32                 2'b00:
33                 begin
34                     currentState <= A;
35                     out <= 1'b0;
36                     change_1 <= 1'b0;
37                     change_2 <= 1'b0;
38                 end
39                 2'b01:
40                 begin
41                     currentState <= B;
42                     out <= 1'b0;
43                     change_1 <= 1'b0;
44                     change_2 <= 1'b0;
45                 end
```

```

46         2'b10:
47             begin
48                 currentState <= C;
49                 out <= 1'b0;
50                 change_1 <= 1'b0;
51                 change_2 <= 1'b0;
52             end
53         2'b11:
54             begin
55                 currentState <= A;
56                 out <= 1'b1;
57                 change_1 <= 1'b0;
58                 change_2 <= 1'b0;
59             end
60         endcase
61         B: case({price_2,price_1})
62         2'b00:
63             begin
64                 currentState <= A;
65                 out <= 1'b0;
66                 change_1 <= 1'b1;
67                 change_2 <= 1'b0;
68             end
69         2'b01:
70             begin
71                 currentState <= C;
72                 out <= 1'b0;
73                 change_1 <= 1'b0;
74                 change_2 <= 1'b0;
75             end
76         2'b10:
77             begin
78                 currentState <= D;
79                 out <= 1'b0;
80                 change_1 <= 1'b0;
81                 change_2 <= 1'b0;
82             end
83         2'b11:
84             begin
85                 currentState <= A;
86                 out <= 1'b1;
87                 change_1 <= 1'b1;
88                 change_2 <= 1'b0;
89             end
90         endcase
91         C: case({price_2,price_1})
92         2'b00:
93             begin
94                 currentState <= A;
95                 out <= 1'b0;
96                 change_1 <= 1'b0;
97                 change_2 <= 1'b1;
98             end

```

```

99      2'b01:
100      begin
101          currentState <= D;
102          out <= 1'b0;
103          change_1 <= 1'b0;
104          change_2 <= 1'b0;
105      end
106      2'b10:
107      begin
108          currentState <= A;
109          out <= 1'b1;
110          change_1 <= 1'b0;
111          change_2 <= 1'b0;
112      end
113      2'b11:
114      begin
115          currentState <= A;
116          out <= 1'b1;
117          change_1 <= 1'b0;
118          change_2 <= 1'b1;
119      end
120      endcase
121      D: case({price_2,price_1})
122      2'b00:
123      begin
124          currentState <= A;
125          out <= 1'b0;
126          change_1 <= 1'b1;
127          change_2 <= 1'b1;
128      end
129      2'b01:
130      begin
131          currentState <= A;
132          out <= 1'b1;
133          change_1 <= 1'b0;
134          change_2 <= 1'b0;
135      end
136      2'b10:
137      begin
138          currentState <= A;
139          out <= 1'b1;
140          change_1 <= 1'b1;
141          change_2 <= 1'b0;
142      end
143      2'b11:
144      begin
145          currentState <= A;
146          out <= 1'b1;
147          change_1 <= 1'b1;
148          change_2 <= 1'b1;
149      end
150      endcase
151      endcase
152      end
153  end
154
155 endmodule

```

```

1 module VendingMachine_15 (clock, reset, price_1, price_2, out, change_1, change_2);
2     input clock;
3     input reset;
4     input price_1, price_2;
5
6     output reg out;
7     output reg change_1, change_2;
8
9     parameter A = 2'b00;
10    parameter B = 2'b01;
11    parameter C = 2'b10;
12
13    reg [1:0] currentState;
14    reg [1:0] nextState;
15
16    always @(posedge clock)
17    begin
18        if (reset)
19        begin
20            currentState <= A;
21            nextState <= A;
22            out <= 1'b0;
23            change_1 <= 1'b0;
24            change_2 <= 1'b0;
25        end
26    else
27        begin
28            currentState <= nextState;
29            case(currentState)
30                A: case({price_2, price_1})
31                    2'b00:
32                    begin
33                        currentState <= A;
34                        out <= 1'b0;
35                        change_1 <= 1'b0;
36                        change_2 <= 1'b0;
37                    end
38                    2'b01:
39                    begin
40                        currentState <= B;
41                        out <= 1'b0;
42                        change_1 <= 1'b0;
43                        change_2 <= 1'b0;
44                    end
45                    2'b10:
46                    begin
47                        currentState <= C;
48                        out <= 1'b0;
49                        change_1 <= 1'b0;
50                        change_2 <= 1'b0;
51                    end
52                    2'b11:
53                    begin
54                        currentState <= A;
55                        out <= 1'b1;
56                        change_1 <= 1'b1;
57                        change_2 <= 1'b0;
58                    end

```



```

59         endcase
60         B: case({price_2,price_1})
61         2'b00:
62             begin
63                 currentState <= A;
64                 out <= 1'b0;
65                 change_1 <= 1'b1;
66                 change_2 <= 1'b0;
67             end
68         2'b01:
69             begin
70                 currentState <= C;
71                 out <= 1'b0;
72                 change_1 <= 1'b0;
73                 change_2 <= 1'b0;
74             end
75         2'b10:
76             begin
77                 currentState <= A;
78                 out <= 1'b1;
79                 change_1 <= 1'b0;
80                 change_2 <= 1'b0;
81             end
82         2'b11:
83             begin
84                 currentState <= A;
85                 out <= 1'b1;
86                 change_1 <= 1'b0;
87                 change_2 <= 1'b1;
88             end
89         endcase
90         C: case({price_2,price_1})
91         2'b00:
92             begin
93                 currentState <= A;
94                 out <= 1'b0;
95                 change_1 <= 1'b0;
96                 change_2 <= 1'b1;
97             end
98         2'b01:
99             begin
100                 currentState <= A;
101                 out <= 1'b1;
102                 change_1 <= 1'b0;
103                 change_2 <= 1'b0;
104             end
105         2'b10:
106             begin
107                 currentState <= A;
108                 out <= 1'b1;
109                 change_1 <= 1'b1;
110                 change_2 <= 1'b0;
111             end
112         2'b11:
113             begin
114                 currentState <= A;
115                 out <= 1'b1;
116                 change_1 <= 1'b1;
117                 change_2 <= 1'b1;
118             end
119         endcase
120     endcase
121 end
122 end
123 endmodule

```

```

1  module VendingMachine_10 (clock, reset, price_1, price_2, out, change_1, change_2);
2      input clock;
3      input reset;
4      input price_1, price_2;
5
6      output reg out;
7      output reg change_1, change_2;
8
9      parameter A = 2'b00;
10     parameter B = 2'b01;
11
12     reg [1:0] currentState;
13     reg [1:0] nextState;
14
15     always @(posedge clock)
16     begin
17         if (reset)
18         begin
19             currentState <= A;
20             nextState <= A;
21             out <= 1'b0;
22             change_1 <= 1'b0;
23             change_2 <= 1'b0;
24         end
25     else
26     begin
27         currentState <= nextState;
28         case(currentState)
29             A: case({price_2, price_1})
30                 2'b00:
31                 begin
32                     currentState <= A;
33                     out <= 1'b0;
34                     change_1 <= 1'b0;
35                     change_2 <= 1'b0;
36                 end
37                 2'b01:
38                 begin
39                     currentState <= B;
40                     out <= 1'b0;
41                     change_1 <= 1'b0;
42                     change_2 <= 1'b0;
43                 end

```

```

44         2'b10:
45             begin
46                 currentState <= A;
47                 out <= 1'b1;
48                 change_1 <= 1'b0;
49                 change_2 <= 1'b0;
50             end
51         2'b11:
52             begin
53                 currentState <= A;
54                 out <= 1'b1;
55                 change_1 <= 1'b0;
56                 change_2 <= 1'b1;
57             end
58     endcase
59     B: case({price_2,price_1})
60     2'b00:
61         begin
62             currentState <= A;
63             out <= 1'b0;
64             change_1 <= 1'b1;
65             change_2 <= 1'b0;
66         end
67     2'b01:
68         begin
69             currentState <= A;
70             out <= 1'b1;
71             change_1 <= 1'b0;
72             change_2 <= 1'b0;
73         end
74     2'b10:
75         begin
76             currentState <= A;
77             out <= 1'b1;
78             change_1 <= 1'b1;
79             change_2 <= 1'b0;
80         end
81     2'b11:
82         begin
83             currentState <= A;
84             out <= 1'b1;
85             change_1 <= 1'b1;
86             change_2 <= 1'b1;
87         end
88     endcase
89 endcase
90 end
91 end
92 endmodule

```

Result:



```
1  module Result (z_10,z_15,z_20,c1_10,c1_15,c1_20,c2_10,c2_15,c2_20,z,c1,c2);
2      input z_10,z_15,z_20,c1_10,c1_15,c1_20,c2_10,c2_15,c2_20;
3
4      output z,c1,c2;
5
6      assign z= z_10 | z_15 | z_20;
7      assign c1= c1_10 | c1_15 | c1_20;
8      assign c2= c2_10 | c2_15 | c2_20;
9
10 endmodule
```

Reset Control:



```
1  module ResetControl (resetin,in,restout);
2      input resetin,in;
3
4      output restout;
5
6      assign restout = resetin | ~in;
7
8  endmodule
```

Priority Encoder:


```
1 module PriorityEncoder (in1, in2, in3, in4, in5, in6, in7, in8, enable, out1, out2, out3);
2
3     input in1, in2, in3, in4, in5, in6, in7, in8;
4     input enable;
5     output reg out1, out2, out3;
6
7     always @ (enable, in1, in2, in3, in4, in5, in6, in7, in8)
8         begin
9             if(enable == 1)
10                begin
11                    if(in8 == 1)    begin out3=1'b1; out2=1'b1; out1=1'b1; end
12                    else if(in7 == 1) begin out3=1'b1; out2=1'b1; out1=1'b0; end
13                    else if(in6 == 1) begin out3=1'b1; out2=1'b0; out1=1'b1; end
14                    else if(in5 == 1) begin out3=1'b1; out2=1'b0; out1=1'b0; end
15                    else if(in4 == 1) begin out3=1'b0; out2=1'b1; out1=1'b1; end
16                    else if(in3 == 1) begin out3=1'b0; out2=1'b1; out1=1'b0; end
17                    else if(in2 == 1) begin out3=1'b0; out2=1'b0; out1=1'b1; end
18                    else if(in1 == 1) begin out3=1'b0; out2=1'b0; out1=1'b0; end
19                    else            begin out3=1'bz; out2=1'bz; out1=1'bz; end
20                end
21                else begin out3=1'bz; out2=1'bz; out1=1'bz; end
22            end
23
24 endmodule
```

Decoder:



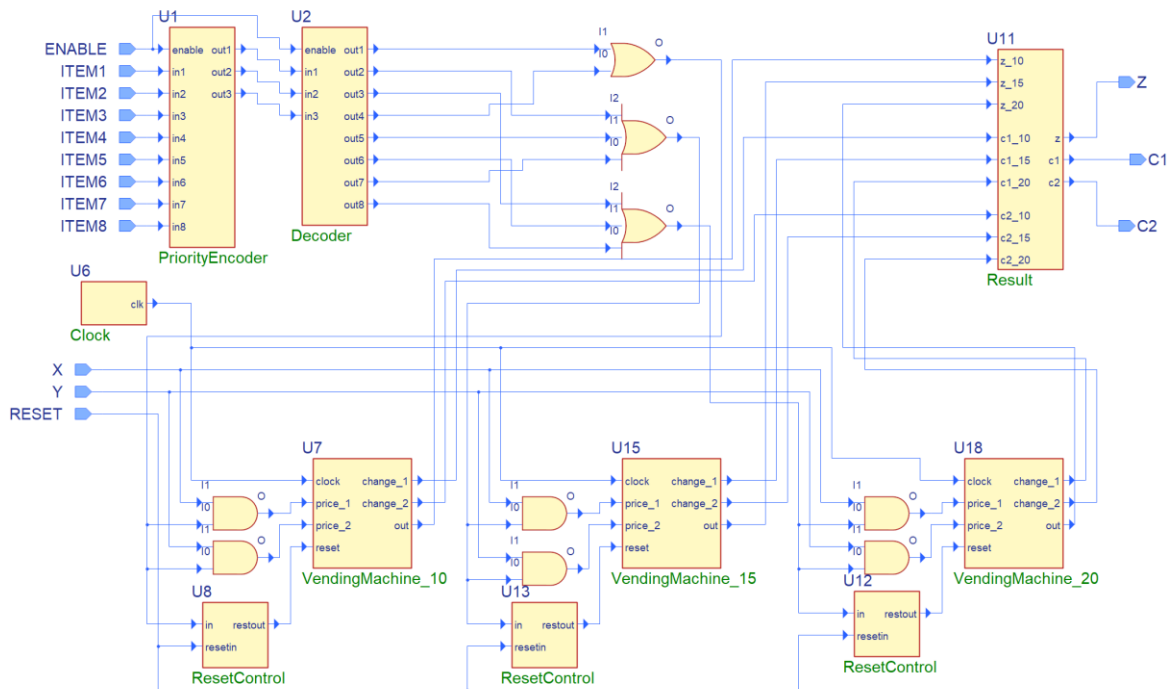
```
1  module Decoder (in1, in2, in3, enable, out1, out2, out3, out4, out5, out6, out7, out8);
2      input in1, in2, in3;
3      input enable;
4
5      output reg out1, out2, out3, out4, out5, out6, out7, out8;
6
7      always @ (enable or in1, in2, in3)
8          begin
9              if(enable == 1)
10                 begin
11                     if(in3==1'b0 && in2==1'b0 && in1==1'b0) begin
12                         {out8,out7,out6,out5,out4,out3,out2,out1} = 8'b0000001; end
13                     else if(in3==1'b0 && in2==1'b0 && in1==1'b1) begin
14                         {out8,out7,out6,out5,out4,out3,out2,out1} = 8'b0000010; end
15                     else if(in3==1'b0 && in2==1'b1 && in1==1'b0) begin
16                         {out8,out7,out6,out5,out4,out3,out2,out1} = 8'b0000100; end
17                     else if(in3==1'b0 && in2==1'b1 && in1==1'b1) begin
18                         {out8,out7,out6,out5,out4,out3,out2,out1} = 8'b0001000; end
19                     else if(in3==1'b1 && in2==1'b0 && in1==1'b0) begin
20                         {out8,out7,out6,out5,out4,out3,out2,out1} = 8'b0010000; end
21                     else if(in3==1'b1 && in2==1'b0 && in1==1'b1) begin
22                         {out8,out7,out6,out5,out4,out3,out2,out1} = 8'b0100000; end
23                     else if(in3==1'b1 && in2==1'b1 && in1==1'b0) begin
24                         {out8,out7,out6,out5,out4,out3,out2,out1} = 8'b1000000; end
25                     else if(in3==1'b1 && in2==1'b1 && in1==1'b1) begin
26                         {out8,out7,out6,out5,out4,out3,out2,out1} = 8'b1000000; end
27                 end
28                 else {out8,out7,out6,out5,out4,out3,out2,out1}=8'bzzzzzzzz;
29             end
30
31
32 endmodule
```

Clock:



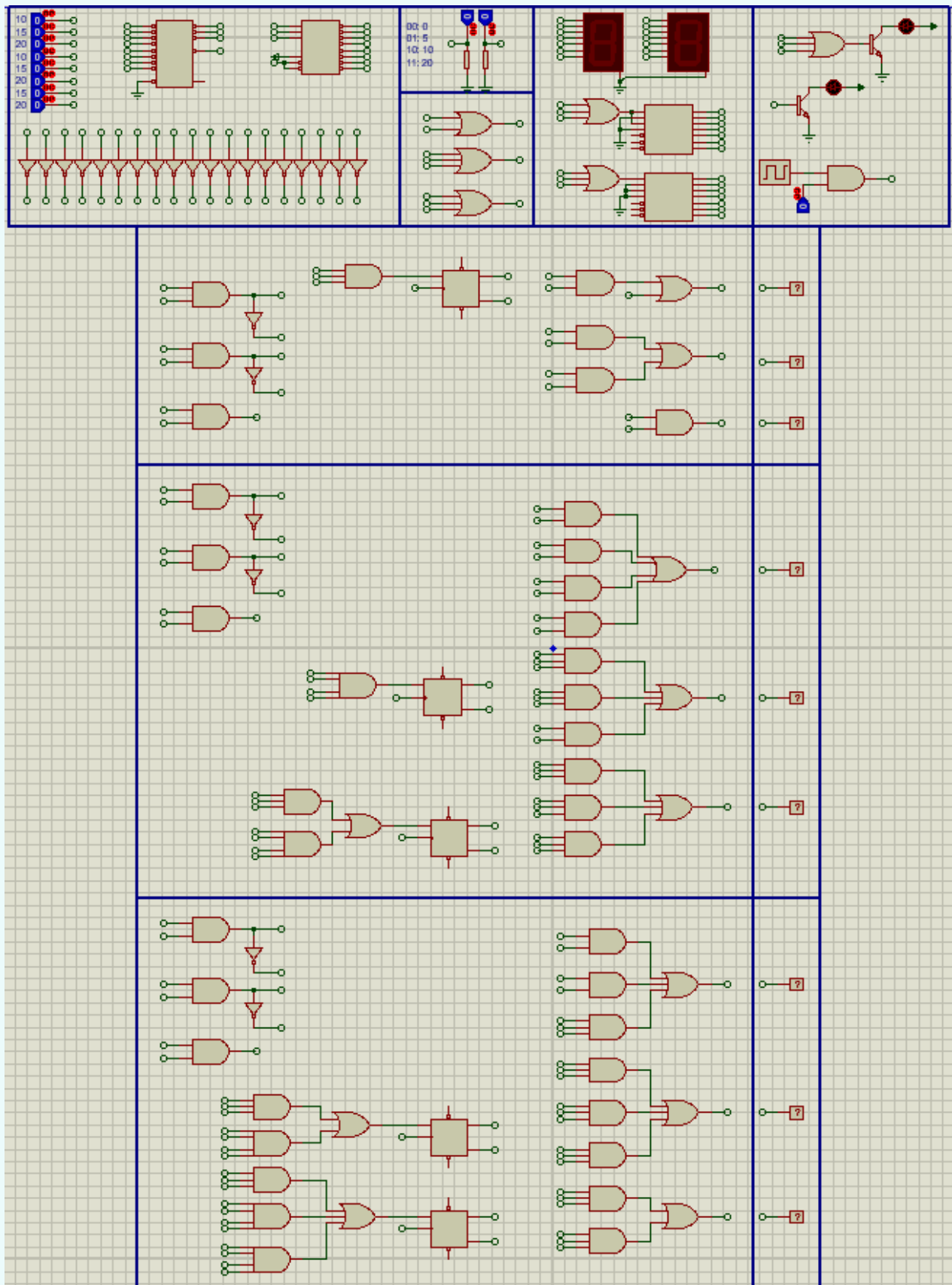
```
1  module Clock (clk);
2      output reg clk;
3
4      initial
5          clk=0;
6
7      always
8          #50 clk = ~clk;
9
10 endmodule
```

Schematic diagram using Active HDL

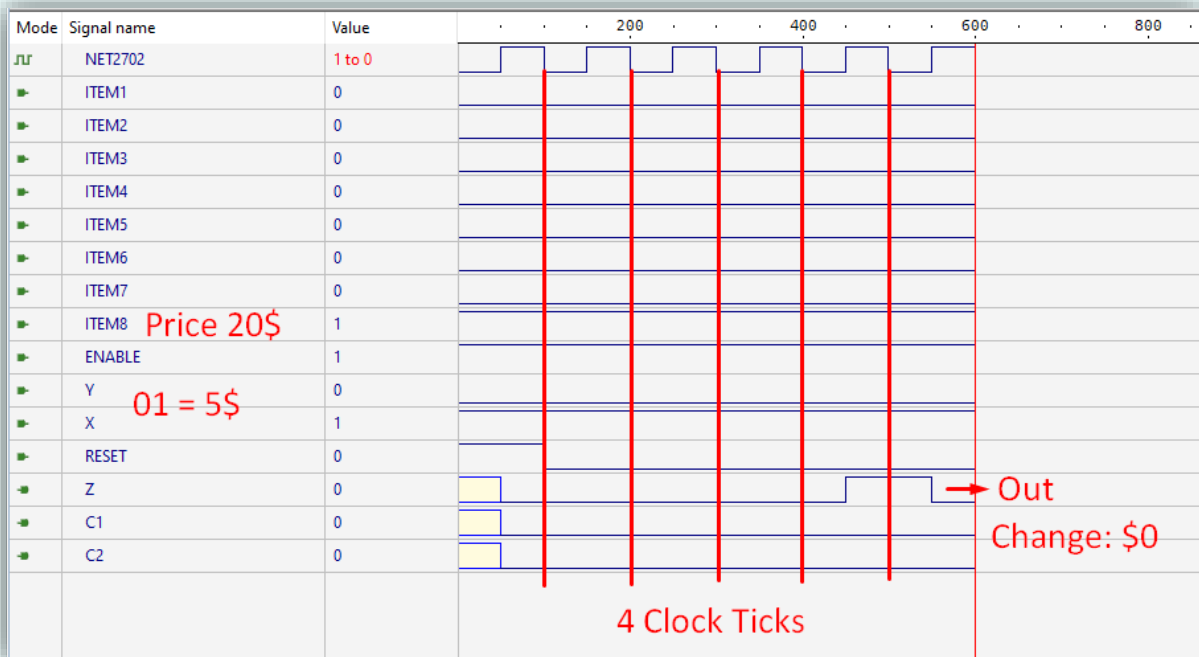
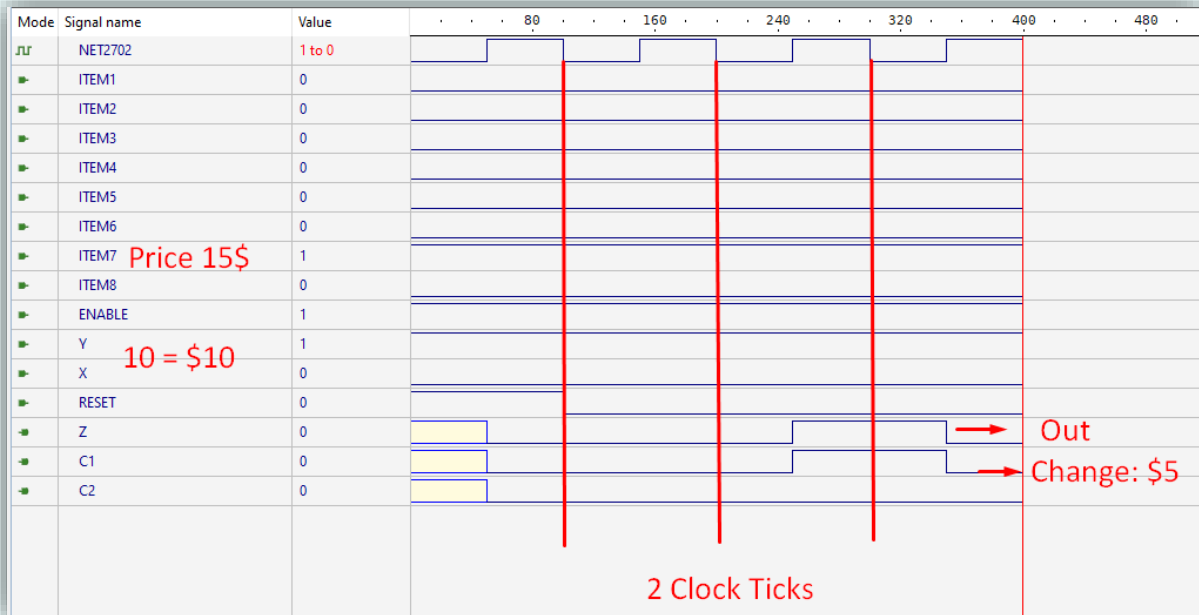


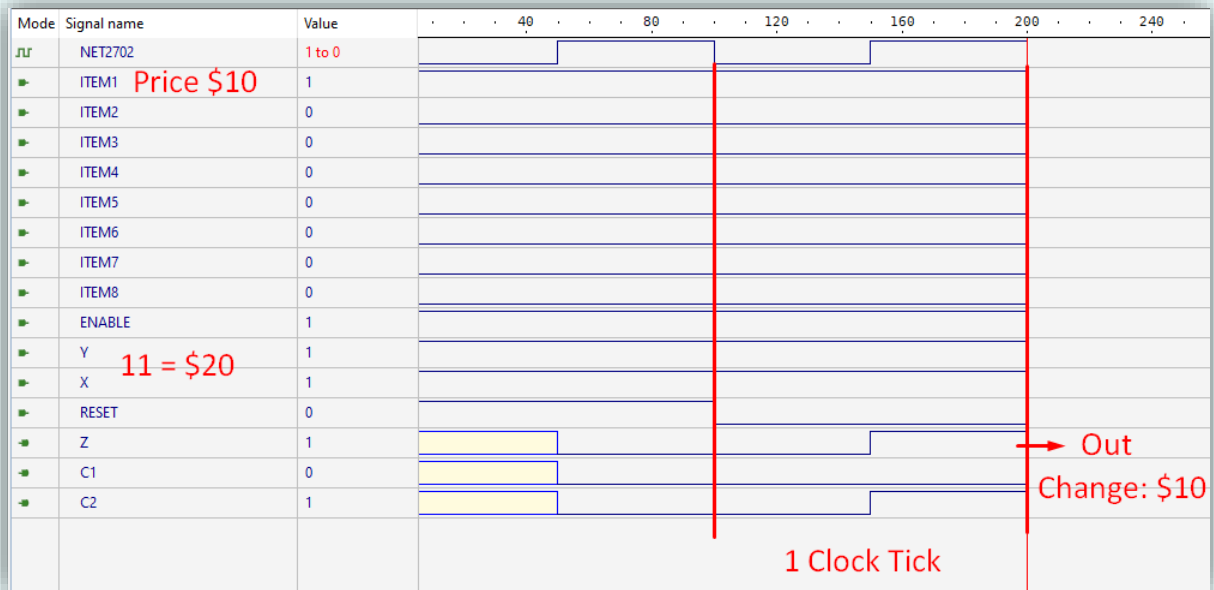
ALDEC	(C)ALDEC, Inc 2260 Corporate Circle Henderson, NV 89074
Created:	12/16/2023
Title:	Vending Machine
Revision:	1.0
Page:	1 / 1

Schematic diagram using Proteus



The design simulation





**THANK
YOU**