به نام خدا

گزارش آزمایش شماره 8

آرشام لولوهری ۹۹۱۰۲۱۵۶
نامی خداداد ۹۹۱۰۱۴۷۳

ماشین قهوه ساز مور

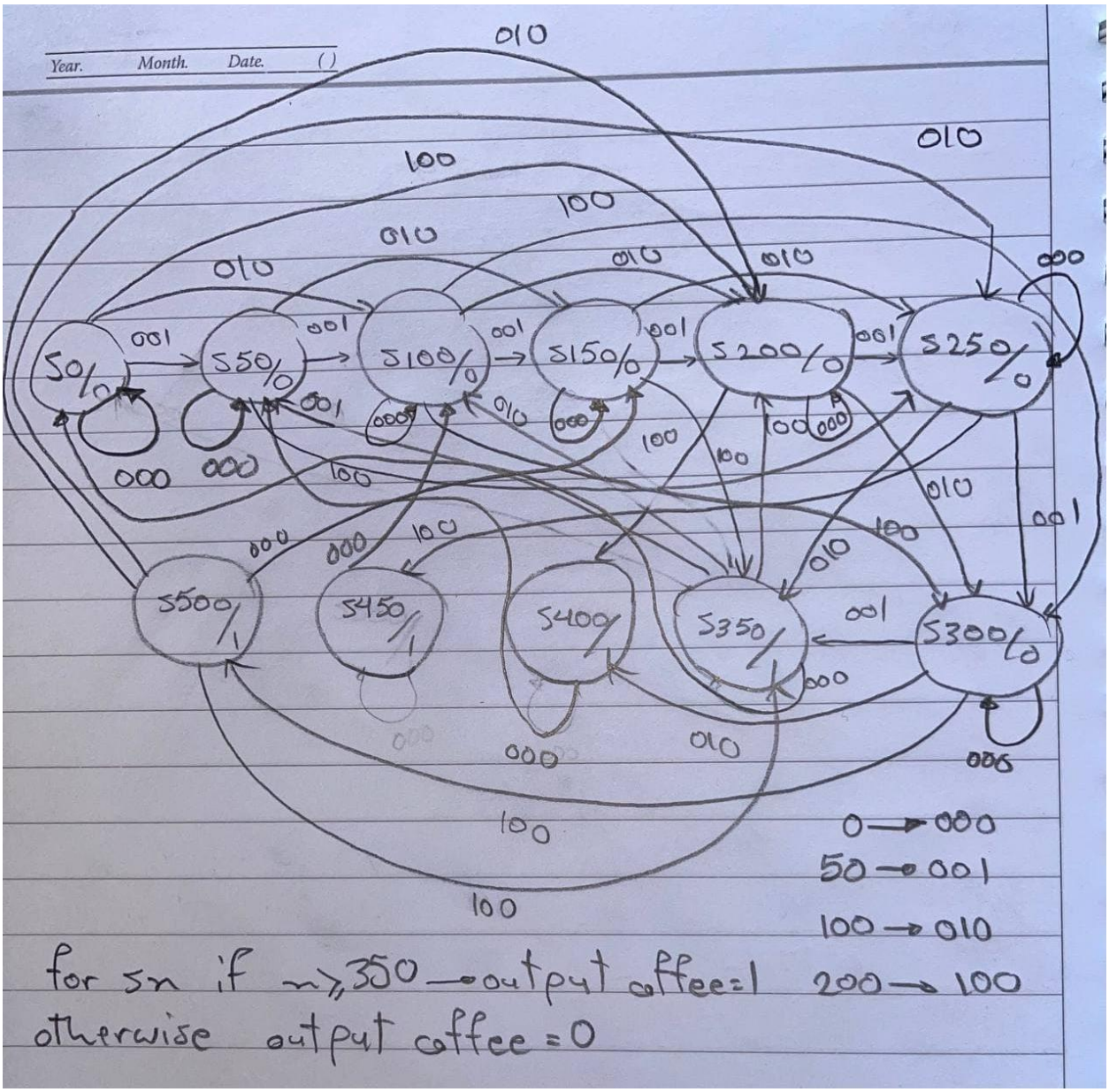
ابتدا باید بدانیم از حالت پول صفر تا حالت ۵۰۰ ، ۱۱ state مختلف داریم:

0,50,100,150,200,250,300,350,400,450,500

که ترتیب از چپ به راست فوق را برای استیت ها به کار می گیریم. ۳ نوع پول ۵۰ ، ۱۰۰ و ۲۰۰ تومنی را هم ملاک قرار میدهیم که به ترتیب ۵۰ را با input ۰۰۱ ، ۱۰۰ را با input ۰۱۰ و ۲۰۰ را با input ۱۰۰ نمایش میدهیم(وارد دستگاه می کنیم).

خروجی هم که سکه های ۵۰ (output 01) و ۱۰۰ (output 10) تومنی، و یک سکه 50 و یک سکه 100 تومنی (output 11)، به همراه وجود (۱) یا عدم وجود (۰) قهوه است.

حال نمودار مور را به شکل زیر برای تفهیم بهتر کد رسم می کنیم:



با توجه به نمودار درباره state ها مطالب زیر را استنباط می کنیم.
اگر state صفر باشیم:
در صورت ورودی صفر، شماره state تغییری نکرده و خروجی صفر است.
ورودی ۵۰ تومان، نیز خروجی صفر است و شماره state یکی به جلو می رود.
ورودی ۱۰۰ تومان و ۲۰۰ تومان نیز به همین شکل خروجی صفر داریم و state ما به ترتیب ۲ و ۴ تا اضافه میشود . در این حالت دستگاه منتظر پول بیشتری می ماند.(پس پولی پس نمی دهد)
اگر state ۵۰ باشد:
در صورت ورودی صفر، شماره state تغییری نکرده و خروجی صفر است.
ورودی ۵۰ تومان، نیز خروجی صفر است و شماره state یکی به جلو می رود.
ورودی ۱۰۰ تومان و ۲۰۰ تومان نیز به همین شکل خروجی صفر داریم و state ما به ترتیب ۲ و ۴ تا اضافه میشود . در این حالت دستگاه منتظر پول بیشتری می ماند.(هنوز پولی پس نمی دهد)
این روند تا state ۳۰۰ دقیقا به همین نحو است.
حال اگر در هر یک از state های 350 تا 500 باشیم، به state ۰ رفته و خروجی ۱ میدهد و بدین ترتیب با خوردن کلاک ، ماشین مور خروجی را تعیین میکند.
در صورت رد کردن ۵۰۰ هم out of range به حساب می آید.
شکل کد و شبیه سازی به صورت ذیل می باشد. توجه داریم که چون ماشین مور است، تعیین خروجی ها و تعیین nextState ها باید در دو بلاک جداگانه انجام شود که بلاک مربوط به تعیین خروجی ها تنها حساس به تغییرات currentState است و به ورودی یا همان money حساس نیست و با تغییر آن اجرا نمیشود:

```
21  module machine_moore(money,coffee,clock,remain,error);
22  input wire [2:0]money;
23  input wire clock;
24  output reg coffee=0;
25  output reg [1:0]remain=0;
26  output reg error=0;
27  parameter s0=3'b000,s50=3'b001,s100=3'b010,s150=3'b011,s200=3'b100,s250=3'b101,s300=3'b110,s350=4'b0111,s400=4'b1000,s450=4'b1001,s500=4'b1010,out_of_range=4'b1111;
28  reg [3:0]nextState = s0;
29  reg [3:0]currentState = s0;
30
31  always@(posedge clock) begin
32  currentState <= nextState;
33  end
34
35  always @(currentState,money)
36  begin
37     if (currentState != out_of_range)
38        error = 0;
39     case(currentState[3:0])
40        s0:
41           begin
42           case(money[2:0])
43           3'b000:
44           begin
45           nextState[3:0]=s0;
46           end
47           3'b001:
48           begin
49           nextState[3:0]=s50;
50           end
51           3'b010:
52           begin
53           nextState[3:0]=s100;
54           end
55           3'b100:
56           begin
57           nextState[3:0]=s200;
58           end
59           endcase
60           end
61        s50:
62           begin
63           case(money[2:0])
64           3'b000:
65           begin
```

```
56          begin
57          nextState[3:0]=s200;
58          end
59          endcase
60          end
61        s50:
62          begin
63          case(money[2:0])
64          3'b000:
65          begin
66          nextState[3:0]=s50;
67          end
68          3'b001:
69          begin
70          nextState[3:0]=s100;
71          end
72          3'b010:
73          begin
74          nextState[3:0]=s150;
75          end
76          3'b100:
77          begin
78          nextState[3:0]=s250;
79          end
80          endcase
81          end
82        s100:
83          begin
84          case(money[2:0])
85          3'b000:
86          begin
87          nextState[3:0]=s100;
88          end
89          3'b001:
90          begin
91          nextState[3:0]=s150;
92          end
93          3'b010:
94          begin
```

```
93          3'b010:
94          begin
95          nextState[3:0]=s200;
96          end
97          3'b100:
98          begin
99          nextState[3:0]=s300;
100         end
101         endcase
102         end
103       s150:
104         begin
105         case(money[2:0])
106         3'b000:
107         begin
108         nextState[3:0]=s150;
109         end
110         3'b001:
111         begin
112         nextState[3:0]=s200;
113         end
114         3'b010:
115         begin
116         nextState[3:0]=s250;
117         end
118         3'b100:
119         begin
120         nextState[3:0]=s0;
121         end
122         endcase
123         end
124       s200:
125         begin
126         case(money[2:0])
127         3'b000:
128         begin
129         nextState[3:0]=s200;
130         end
131         3'b001:
```

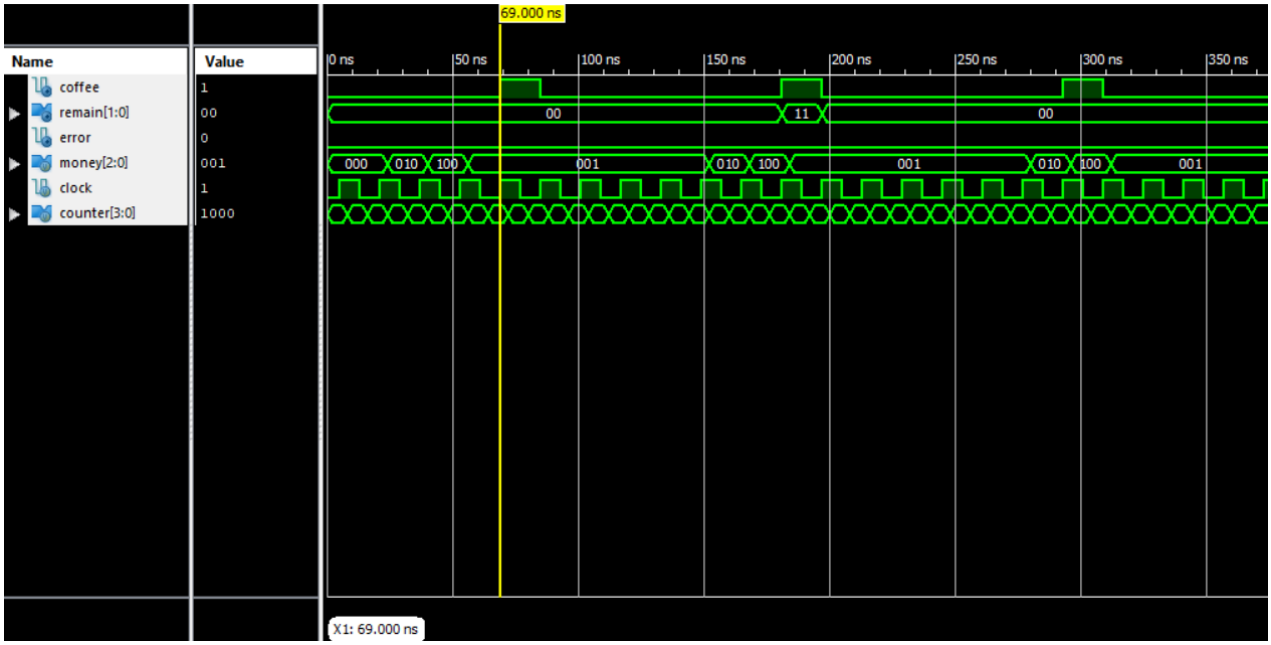
```
130         end
131         3'b001:
132         begin
133         nextState[3:0]=s250;
134         end
135         3'b010:
136         begin
137         nextState[3:0]=s300;
138         end
139         3'b100:
140         begin
141         nextState[3:0]=s400;
142         end
143         endcase
144         end
145       s250:
146         begin
147         case(money[2:0])
148         3'b000:
149         begin
150         nextState[3:0]=s250;
151         end
152         3'b001:
153         begin
154         nextState[3:0]=s300;
155         end
156         3'b010:
157         begin
158         nextState[3:0]=s350;
159         end
160         3'b100:
161         begin
162         nextState[3:0]=s450;
163         end
164         endcase
165         end
166       s300:
167         begin
168         case(money[2:0])
```

```
165         end
166       s300:
167         begin
168         case(money[2:0])
169         3'b000:
170         begin
171         nextState[3:0]=s300;
172         end
173         3'b001:
174         begin
175         nextState[3:0]=s350;
176         end
177         3'b010:
178         begin
179         nextState[3:0]=s400;
180         end
181         3'b100:
182         begin
183         nextState[3:0]=s500;
184         end
185         endcase
186         end
187
188         s350:
189         begin
190            nextState = s0;
191         end
192
193         s400:
194         begin
195            nextState = s0;
196         end
197
198         s450:
199         begin
200            nextState = s0;
201         end
202
203         s500:
```

```
203             s500:
204             begin
205                 nextState = s0;
206             end
207
208         endcase
209     end
210
211
212     always @(currentState)
213       begin
214         case(currentState)
215             s0:
216                 begin
217                     coffee=0;
218                     remain=2'b00;
219                 end
220             s50:
221                 begin
222                     coffee=0;
223                     remain=2'b00;
224                 end
225             s100:
226                 begin
227                     coffee=0;
228                     remain=2'b00;
229                 end
230             s150:
231                 begin
232                     coffee=0;
233                     remain=2'b00;
234                 end
235             s200:
236                 begin
237                     coffee=0;
238                     remain=2'b00;
239                 end
240             s250:
241                 begin
```

```
            end
         s250:
            begin
               coffee=0;
               remain=2'b00;
            end
         s300:
            begin
               coffee=0;
               remain=2'b00;
            end
            s350:
            begin
               coffee = 1;
               remain = 2'b00;
            end

            s400:
            begin
               coffee = 1;
               remain = 2'b01;
            end

            s450:
            begin
               coffee = 1;
               remain = 2'b10;
            end

            s500:
            begin
               coffee = 1;
               remain = 2'b11;
            end

      endcase
   end

   endmodule
```

```
      // Initialize Inputs
      money = 0;
      clock = 0;

      // Wait 100 ns for global reset to finish
      #100;

      // Add stimulus here

   end


      reg[3:0] counter = 4'd0;

      always begin
      #5
         clock = ~clock;
         #3
         counter = counter+1;

         if (counter == 4'd3)
            money = 3'b010;
         else if (counter == 4'd5)
            money = 3'b100;
         else if (counter == 4'd7)
            money = 3'b001;
      end



endmodule
```



ماشین قهوه ساز میلی

این بار نیز مانند قسمت قبل مقدمات را میچینیم:

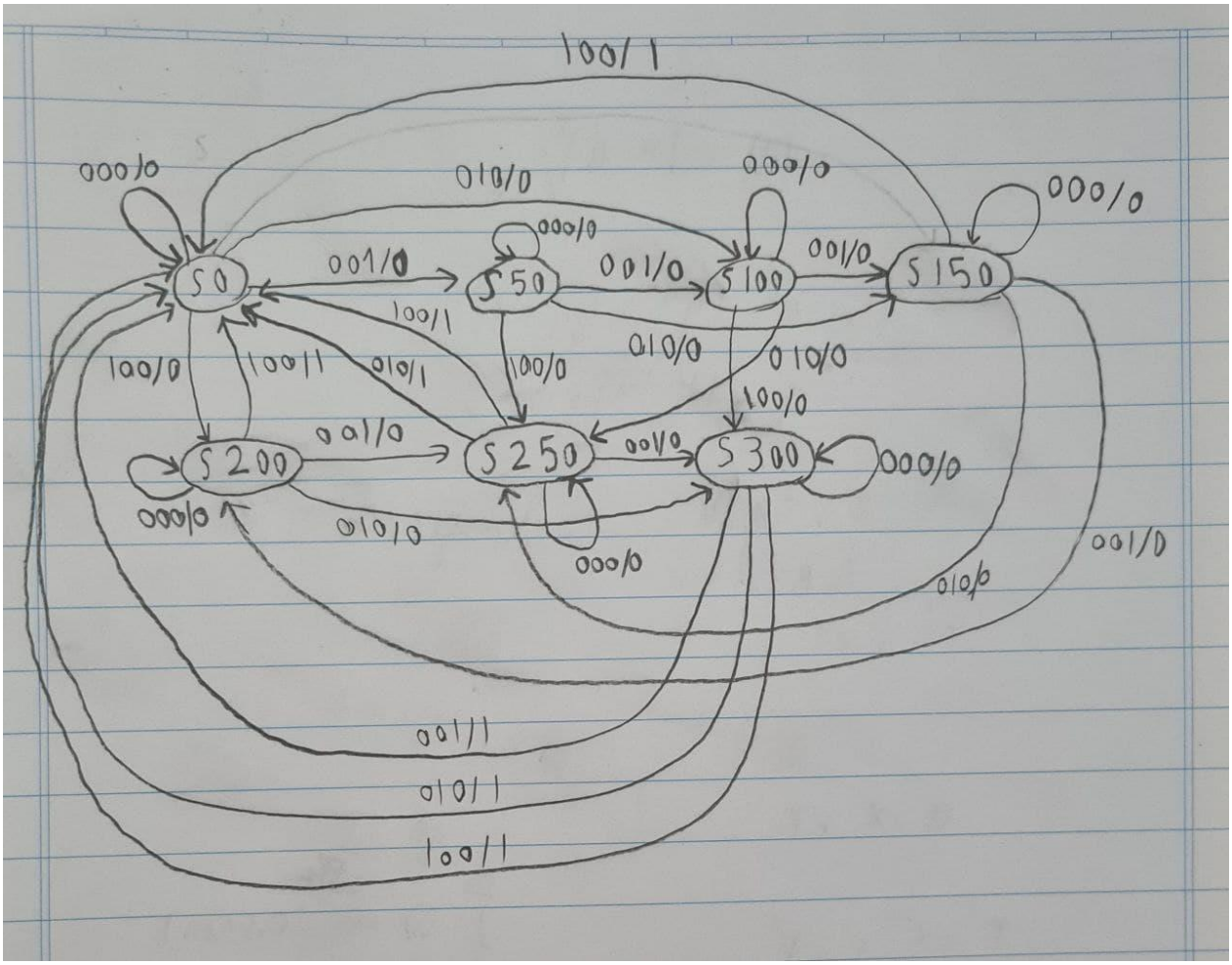
ابتدا باید بدانیم از حالت پول صفر تا حالت 300 ، 7 state مختلف داریم:

0,50,100,150,200,250,300

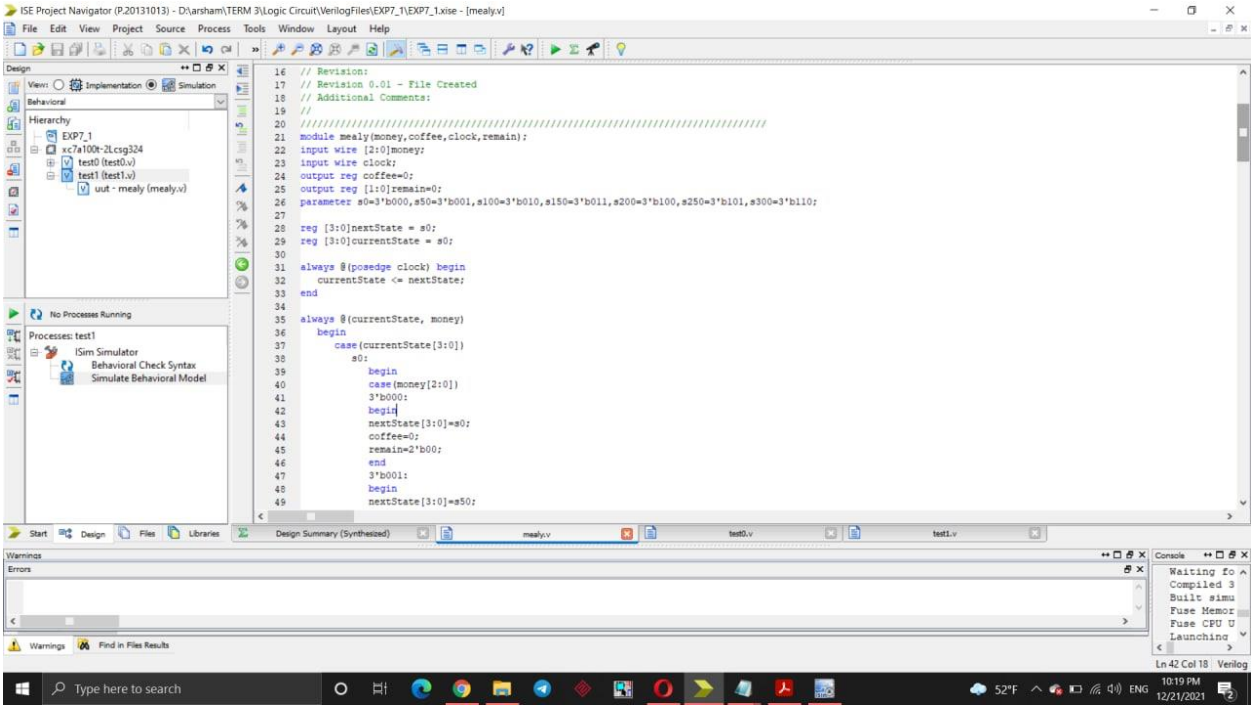
که ترتیب از چپ به راست فوق را برای استیت ها به کار می گیریم. ۳ نوع پول ۵۰ ، ۱۰۰ و ۲۰۰ تومنی را هم ملاک قرار میدهیم که به ترتیب ۵۰ را با input ۰۰۱ ، ۱۰۰ را با input ۰۱۰ و ۲۰۰ را با input ۱۰۰ نمایش میدهیم(وارد دستگاه می کنیم).

خروجی هم که سکه های ۵۰ (output 01) و ۱۰۰ (output 10) تومنی ، و یک سکه 50 و یک سکه 100 تومنی (output 11)، به همراه وجود (۱) یا عدم وجود (۰) قهوه است.

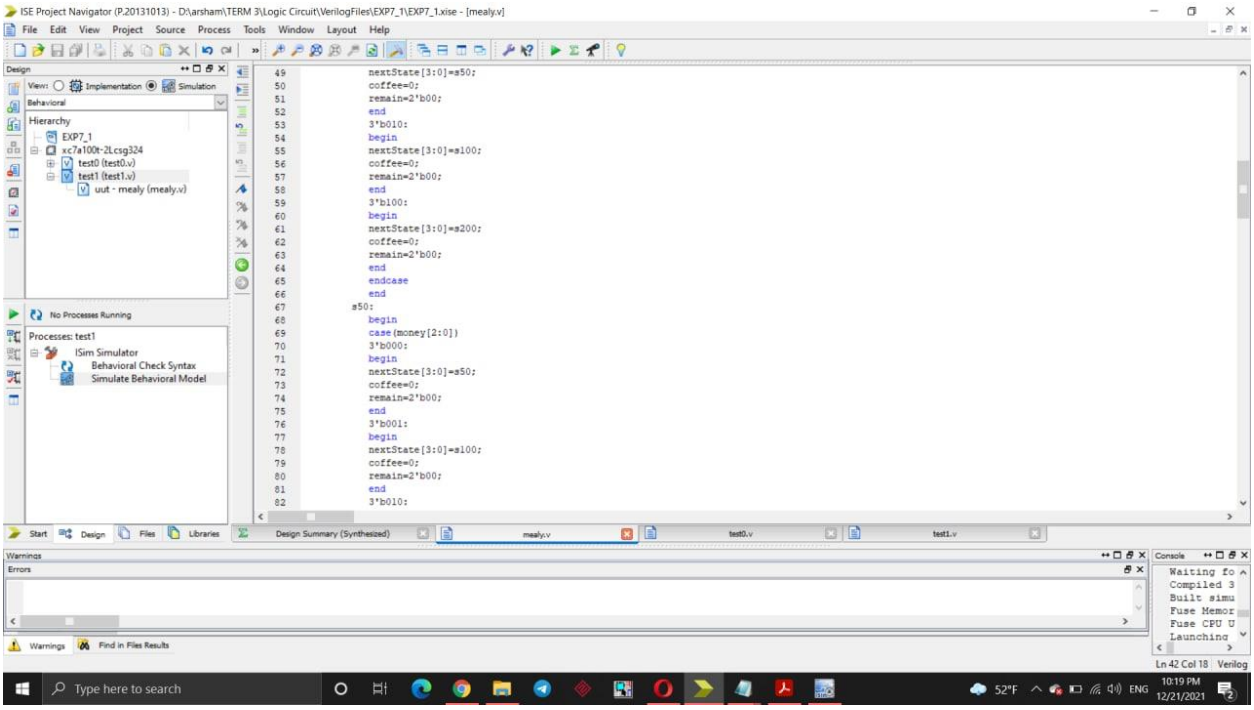
حال نمودار میلی را به شکل زیر برای تفهیم بهتر کد رسم می کنیم:



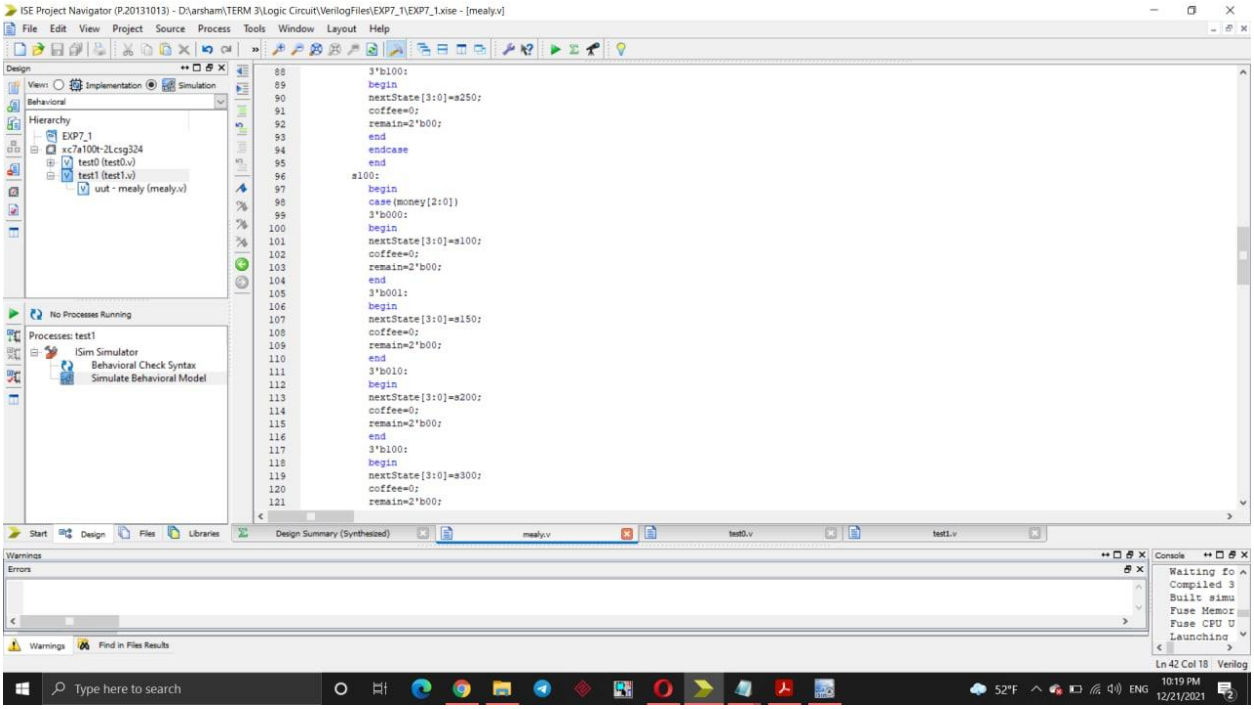
توجه کنید که اینبار 300 state حد حالت ماست و به محض اینکه در مجموع ورودی و state موجود ، به عددی بزرگتر از 300 برسیم ، بدون منتظر ماندن برای کلاک بعدی ، خروجی coffee یک شده و قهوه میدهد و بسته به مقدار پول دریافت شده ، باقیمانده را پس داده و به state صفر برمیگردد.
مانند قسمت قبل با اضافه شدن مبالغ ۵۰ ، ۱۰۰ و ۲۰۰ تومنی به ترتیب یک ، دو و چهار state اضافه می شوند و اگر مرز ۳۵۰ را رد کنند ، خروجی یک میدهند و باقی پول به صورت سکه ۵۰ یا ۱۰۰ تومنی پس داده میشود و سپس به state صفر باز میگردد.
شکل کد و شبیه سازی به صورت ذیل است. توجه داریم که چون ماشین میلی است ، هم خروجی ها و هم nextState داخل یک بلاک قرار دارند که این بلاک به ورودی money نیز حساس است ، یعنی با گرفتن ورودی نیز اجرا میشد و در نتیجه دیگر منتظر کلاک بعدی نمیماند و با توجه به currentState و ورودی در همان لحظه ، خروجی را مشخص میکند:

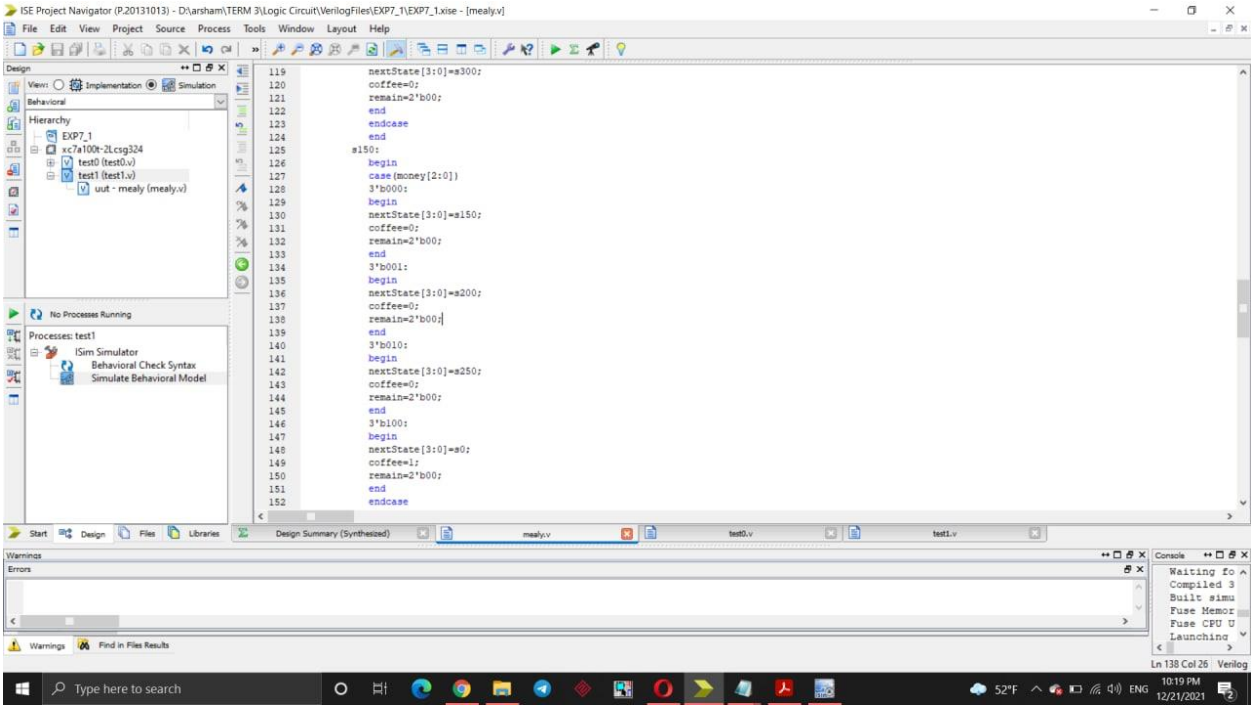


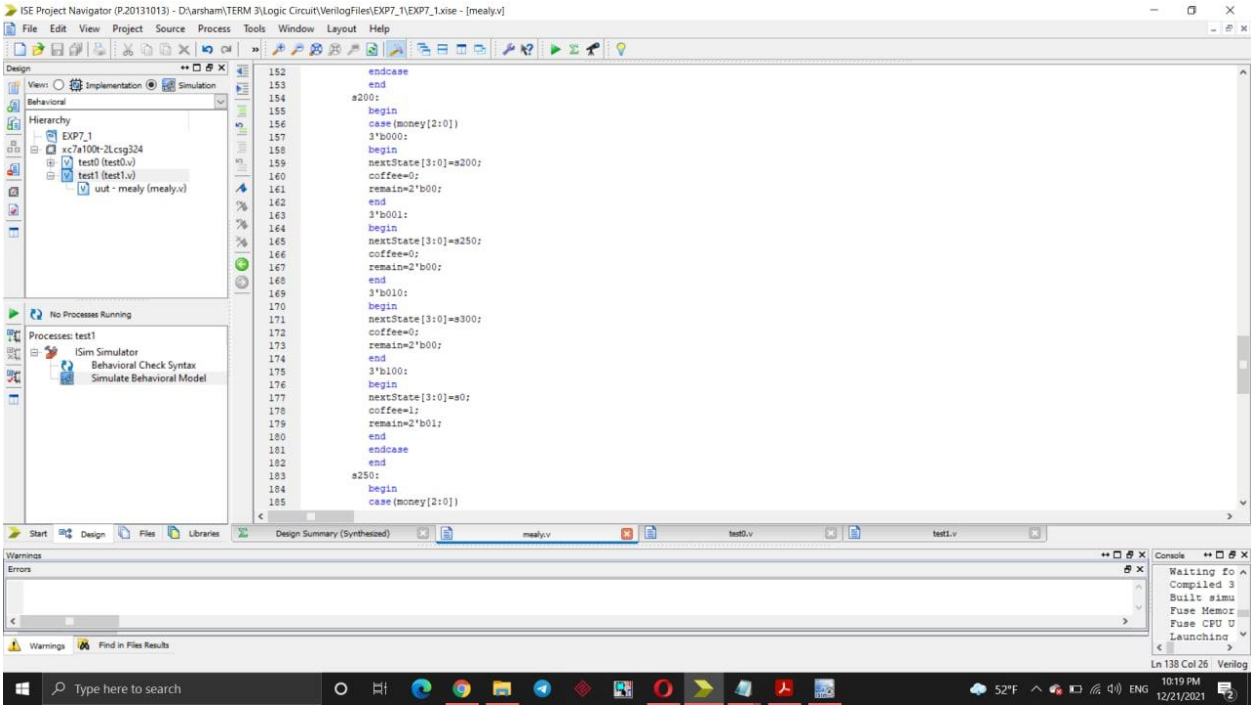

```
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21  module mealy(money,coffee,clock,remain);
22  input wire [2:0]money;
23  input wire clock;
24  output reg coffee=0;
25  output reg [1:0]remain=0;
26  parameter s0=3'b000,s50=3'b001,s100=3'b010,s150=3'b011,s200=3'b100,s250=3'b101,s300=3'b110;
27
28  reg [3:0]nextState = s0;
29  reg [3:0]currentState = s0;
30
31  always @(posedge clock) begin
32     currentState <= nextState;
33  end
34
35  always @(currentState, money)
36     begin
37        case(currentState[3:0])
38           s0:
39              begin
40              case(money[2:0])
41              3'b000:
42              begin
43              nextState[3:0]=s0;
44              coffee=0;
45              remain=2'b00;
46              end
47              3'b001:
48              begin
49              nextState[3:0]=s50;
```

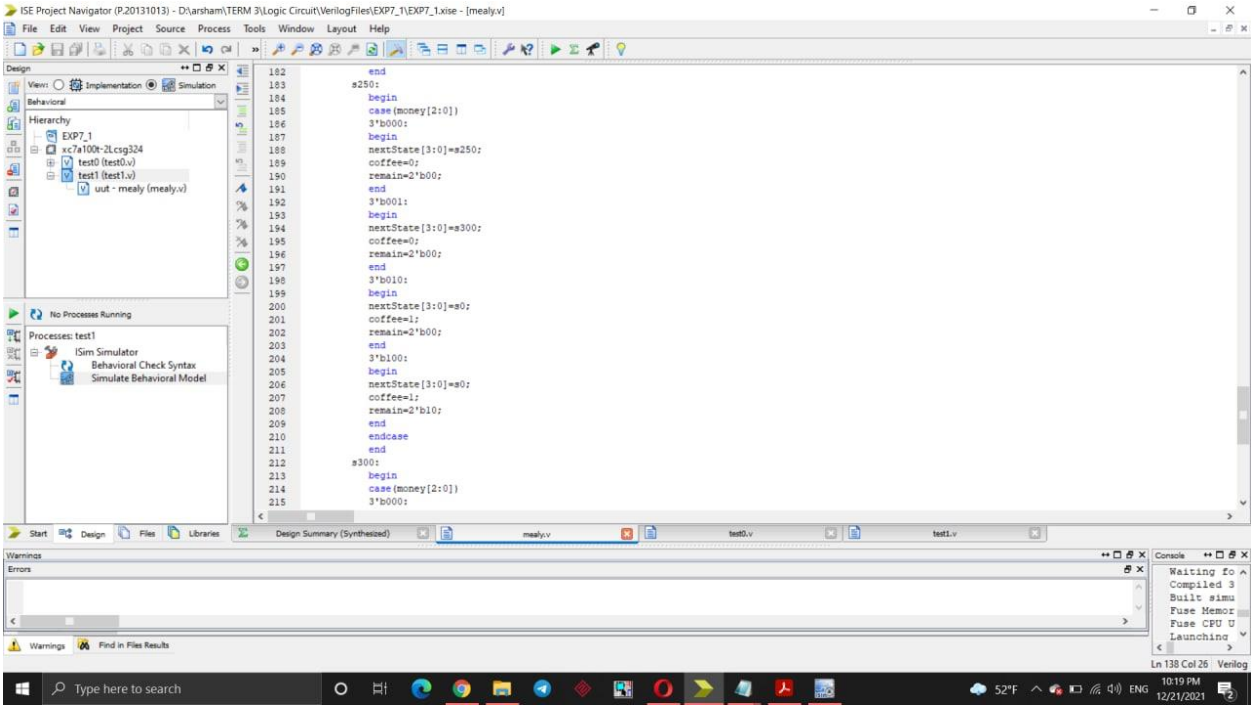


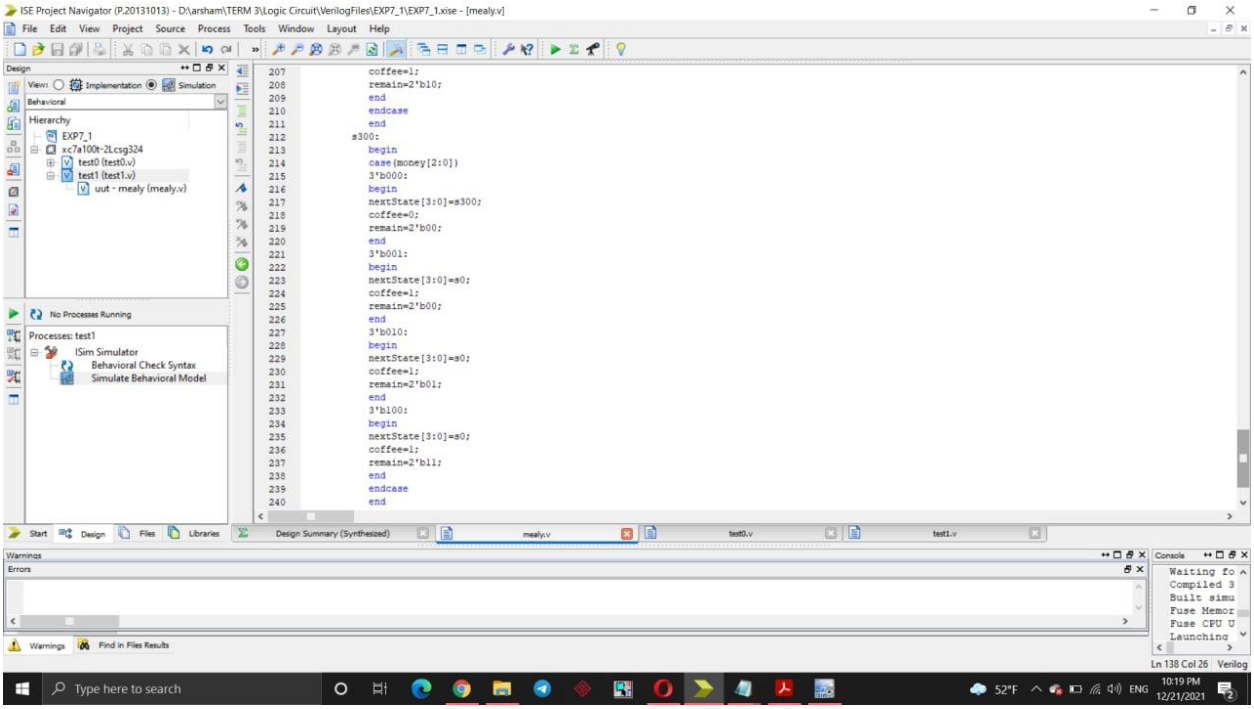

```
49              nextState[3:0]=s50;
50              coffee=0;
51              remain=2'b00;
52              end
53              3'b010:
54              begin
55              nextState[3:0]=s100;
56              coffee=0;
57              remain=2'b00;
58              end
59              3'b100:
60              begin
61              nextState[3:0]=s200;
62              coffee=0;
63              remain=2'b00;
64              end
65              endcase
66              end
67           s50:
68              begin
69              case(money[2:0])
70              3'b000:
71              begin
72              nextState[3:0]=s50;
73              coffee=0;
74              remain=2'b00;
75              end
76              3'b001:
77              begin
78              nextState[3:0]=s100;
79              coffee=0;
80              remain=2'b00;
81              end
82              3'b010:
```

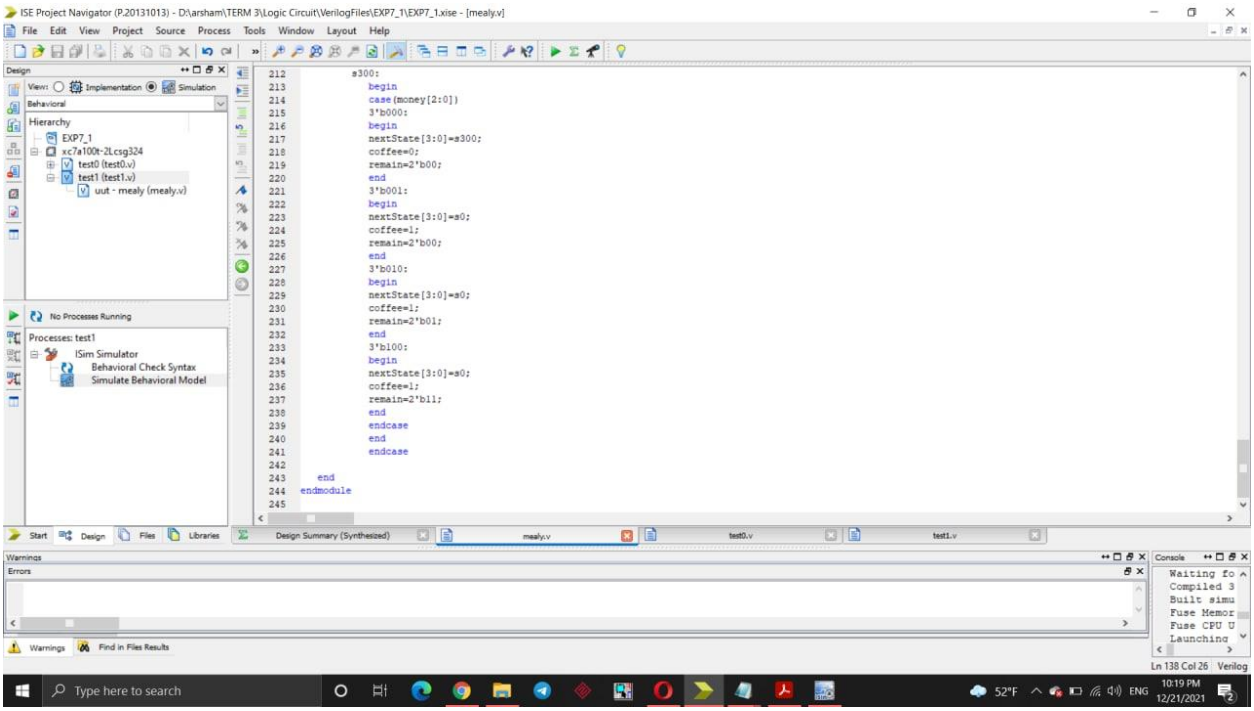

ISE Project Navigator (P.20131013) - D:\arsham\TERM 3\Logic Circuit\VerilogFiles\EXP7_1\EXP7_1.xise - [mealy.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
EXP7_1
xc7a100t-2Lcsg324
test0 (test0.v)
test1 (test1.v)
uut - mealy (mealy.v)
No Processes Running
Processes: test1
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
3'b100:
begin
nextState[3:0]=s250;
coffee=0;
remain=2'b00;
end
endcase
end
s100:
begin
case(money[2:0])
3'b000:
begin
nextState[3:0]=s100;
coffee=0;
remain=2'b00;
end
3'b001:
begin
nextState[3:0]=s150;
coffee=0;
remain=2'b00;
end
3'b010:
begin
nextState[3:0]=s200;
coffee=0;
remain=2'b00;
end
3'b100:
begin
nextState[3:0]=s300;
coffee=0;
remain=2'b00;
Start Design Files Libraries
Design Summary (Synthesized)
mealy.v
test0.v
test1.v
Warnings
Errors
Warnings Find in Files Results
Console
Waiting fo
Compiled 3
Built simu
Fuse Memor
Fuse CPU U
Launching
Ln 42 Col 18 Verilog
Type here to search
52°F
ENG
10:19 PM
12/21/2021

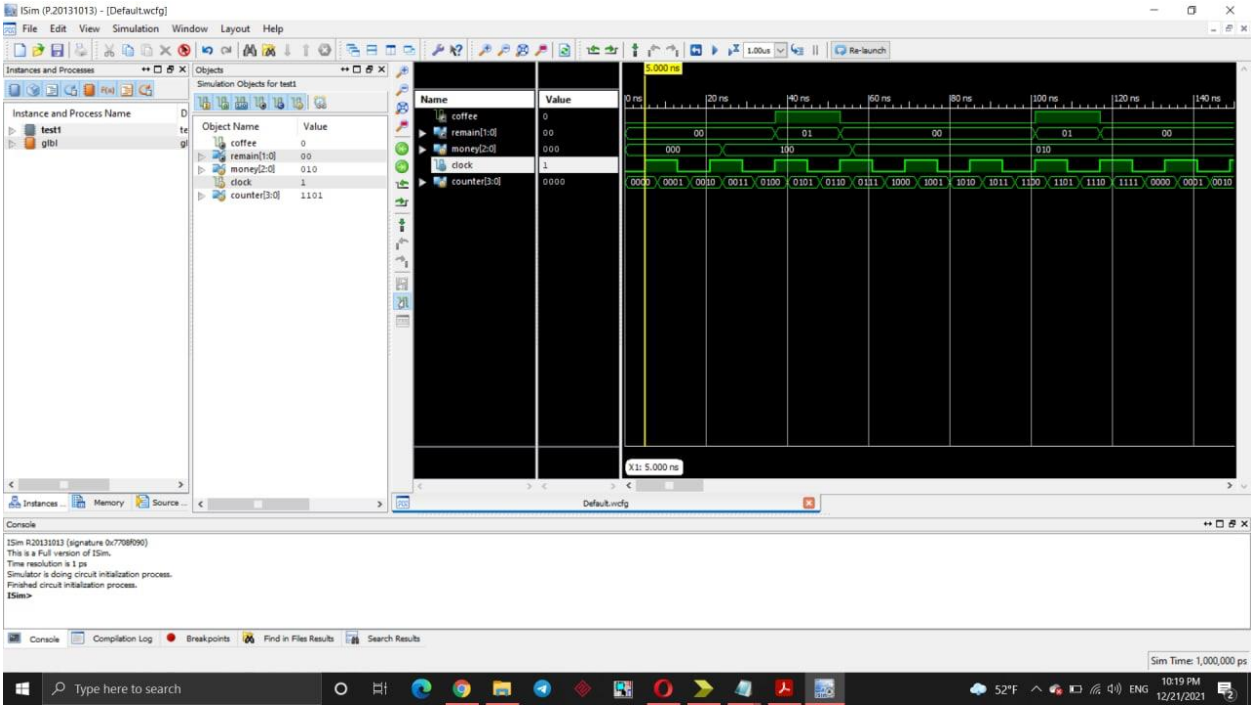

ISE Project Navigator (P.20131013) - D:\arsham\TERM 3\Logic Circuit\VerilogFiles\EXP7_1\EXP7_1.xise - [mealy.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
EXP7_1
xc7a100t-2Lcsg324
test0 (test0.v)
test1 (test1.v)
uut - mealy (mealy.v)
No Processes Running
Processes: test1
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
nextState[3:0]=s300;
coffee=0;
remain=2'b00;
end
endcase
end
s150:
begin
case(money[2:0])
3'b000:
begin
nextState[3:0]=s150;
coffee=0;
remain=2'b00;
end
3'b001:
begin
nextState[3:0]=s200;
coffee=0;
remain=2'b00;
end
3'b010:
begin
nextState[3:0]=s250;
coffee=0;
remain=2'b00;
end
3'b100:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b00;
end
endcase
Start Design Files Libraries
Design Summary (Synthesized)
mealy.v
test0.v
test1.v
Warnings
Errors
Warnings Find in Files Results
Console
Waiting fo
Compiled 3
Built simu
Fuse Memor
Fuse CPU U
Launching
Ln 138 Col 26 Verilog
Type here to search
52°F
ENG
10:19 PM
12/21/2021


ISE Project Navigator (P.20131013) - D:\arsham\TERM 3\Logic Circuit\VerilogFiles\EXP7_1\EXP7_1.xise - [mealy.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
EXP7_1
xc7a100t-2Lcsg324
test0 (test0.v)
test1 (test1.v)
uut - mealy (mealy.v)
No Processes Running
Processes: test1
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
endcase
end
s200:
begin
case(money[2:0])
3'b000:
begin
nextState[3:0]=s200;
coffee=0;
remain=2'b00;
end
3'b001:
begin
nextState[3:0]=s250;
coffee=0;
remain=2'b00;
end
3'b010:
begin
nextState[3:0]=s300;
coffee=0;
remain=2'b00;
end
3'b100:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b01;
end
endcase
end
s250:
begin
case(money[2:0])
Start Design Files Libraries
Design Summary (Synthesized)
mealy.v
test0.v
test1.v
Warnings
Errors
Warnings Find in Files Results
Console
Waiting fo
Compiled 3
Built simu
Fuse Memor
Fuse CPU U
Launching
Ln 138 Col 26 Verilog
Type here to search
52°F
ENG
10:19 PM
12/21/2021


ISE Project Navigator (P.20131013) - D:\arsham\TERM 3\Logic Circuit\VerilogFiles\EXP7_1\EXP7_1.xise - [mealy.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
EXP7_1
xc7a100t-2Lcsg324
test0 (test0.v)
test1 (test1.v)
uut - mealy (mealy.v)
No Processes Running
Processes: test1
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
end
s250:
begin
case(money[2:0])
3'b000:
begin
nextState[3:0]=s250;
coffee=0;
remain=2'b00;
end
3'b001:
begin
nextState[3:0]=s300;
coffee=0;
remain=2'b00;
end
3'b010:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b00;
end
3'b100:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b10;
end
endcase
end
s300:
begin
case(money[2:0])
3'b000:
Start Design Files Libraries
Design Summary (Synthesized)
mealy.v
test0.v
test1.v
Warnings
Errors
Warnings Find in Files Results
Console
Waiting fo
Compiled 3
Built simu
Fuse Memor
Fuse CPU U
Launching
Ln 138 Col 26 Verilog
Type here to search
52°F
ENG
10:19 PM
12/21/2021


ISE Project Navigator (P.20131013) - D:\arsham\TERM 3\Logic Circuit\VerilogFiles\EXP7_1\EXP7_1.xise - [mealy.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
EXP7_1
xc7a100t-2Lcsg324
test0 (test0.v)
test1 (test1.v)
uut - mealy (mealy.v)
No Processes Running
Processes: test1
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
coffee=1;
remain=2'b10;
end
endcase
end
s300:
begin
case(money[2:0])
3'b000:
begin
nextState[3:0]=s300;
coffee=0;
remain=2'b00;
end
3'b001:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b00;
end
3'b010:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b01;
end
3'b100:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b11;
end
endcase
end
Start Design Files Libraries
Design Summary (Synthesized)
mealy.v
test0.v
test1.v
Warnings
Errors
Console
Waiting fo
Compiled 3
Built simu
Fuse Memor
Fuse CPU U
Launching
Warnings Find in Files Results
Ln 138 Col 26 Verilog
Type here to search
52°F
ENG
10:19 PM
12/21/2021


ISE Project Navigator (P.20131013) - D:\arsham\TERM 3\Logic Circuit\VerilogFiles\EXP7_1\EXP7_1.xise - [mealy.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
EXP7_1
xc7a100t-2Lcsg324
test0 (test0.v)
test1 (test1.v)
uut - mealy (mealy.v)
No Processes Running
Processes: test1
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
s300:
begin
case(money[2:0])
3'b000:
begin
nextState[3:0]=s300;
coffee=0;
remain=2'b00;
end
3'b001:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b00;
end
3'b010:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b01;
end
3'b100:
begin
nextState[3:0]=s0;
coffee=1;
remain=2'b11;
end
endcase
end
endcase
end
endmodule
Start Design Files Libraries
Design Summary (Synthesized)
mealy.v
test0.v
test1.v
Warnings
Errors
Console
Waiting fo
Compiled 3
Built simu
Fuse Memor
Fuse CPU U
Launching
Warnings Find in Files Results
Ln 138 Col 26 Verilog
Type here to search
52°F
ENG
10:19 PM
12/21/2021


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Re-launch
Instances and Processes
Instance and Process Name
test1
glbl
Objects
Simulation Objects for test1
Object Name Value
coffee 0
remain[1:0] 00
money[2:0] 010
clock 1
counter[3:0] 1101
Name Value
coffee 0
remain[1:0] 00
money[2:0] 000
clock 1
counter[3:0] 0000
5,000 ns
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
X1: 5,000 ns
Instances
Memory
Source
Default.wcfg
Console
ISim R20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 1,000,000 ps
Type here to search
52°F
ENG
10:19 PM
12/21/2021

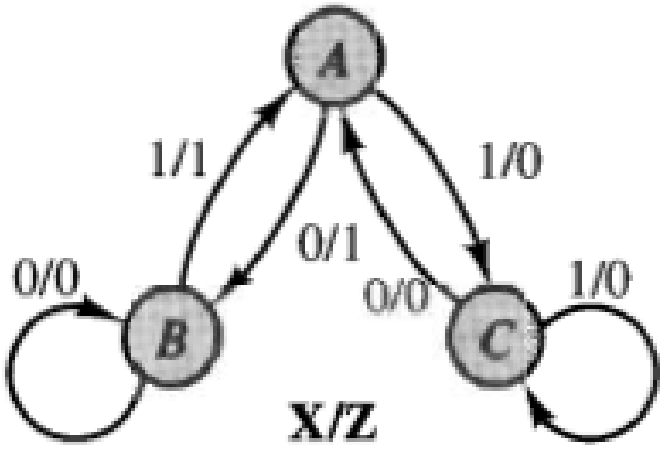
پرسش ها:

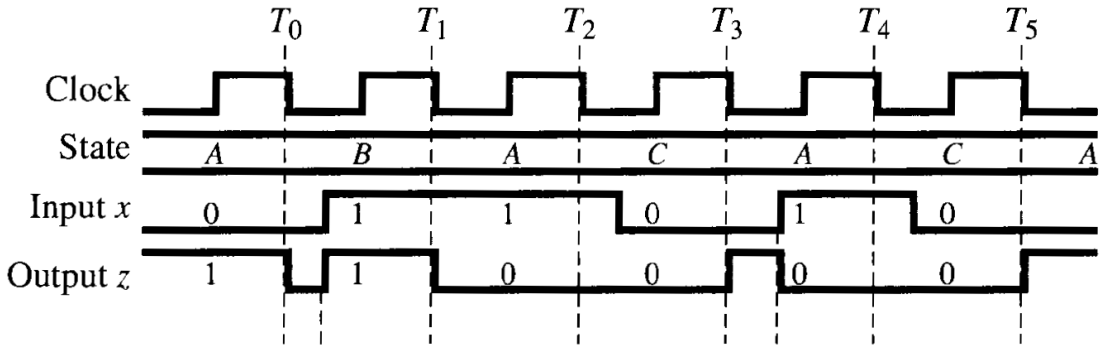
2.1:

دیاگرام سمت راست ، مور و دیاگرام سمت چپ میلی است.
در دیاگرام سمت راست مشاهده میشود که روی پیکان های انتقال حالت یا همان state ، تنها ورودی نوشته شده است. این مشخص میکند که اگر ورودی ای را در یک استیت خاص داشته باشیم و به لبه کلاک (بسته به نوع ماشین ، لبه مثبت یا منفی) برسیم ، استیت ما به چه حالتی تغییر میکند و یا اینکه ثابت میماند. اما در داخل هر استیت ، رقم سمت راست خروجی را نشان داده است. بدین معنا که خروجی ، مستقیما با استیتی که در آن قرار داریم تعیین میشود و ورودی ارتباط مستقیم با خروجی ندارد و صرفا میتواند پس از هر کلاک ، وضعیت استیت را تغییر دهد. پس ماشین مور است.
در دیاگرام سمت چپ خروجی در سمت راست بیت ورودی و روی پیکان ها نوشته شده است. در این حالت اگر روی استیت خاصی باشیم و ورودی معینی داشته باشیم ، آن ورودی فورا میتواند خروجی را تغییر دهد و منتظر کلاک نمیماند. هرچند که تغییر وضعیت استیت همچنان پس از کلاک خوردن رخ میدهد. پس در این حالت ، خروجی تابع مستقیم ورودی و نیز استیت است که نشان میدهد ماشین میلی داریم.

2.2:

در حالت کلی ، معمولا ماشین مور نسبت به میلی مرجح است.برای توجیه این موضوع ، فرض میکنیم یک ماشین میلی با دیاگرام حالت و زمان زیر داریم:





اگر به نحوه ایجاد خروجی با توجه به ورودی های داده شده نگاه کنیم ، میبینیم در بازه ای بین T0,T1 و نیز در بازه ای بین T3,T4 ، خروجی نسبت به قبل و بعد آن متفاوت است. این موضوع بدلیل تابعیت مستقیم خروجی از ورودی است که باعث میشود بلافاصله پس از تغییر ورودی و بدون اینکه مدار برای کلاک بعدی صبر کند ، خروجی تغییر کند. این موضوع میتواند باعث ایجاد glitch های ناخواسته ای در مدل میلی شود که اگر در نظر گرفته نشوند ممکن است عواقب بدی به بار بیاورند. در حالی که ماشین مور تابع مستقیم ورودی نیست و تمام تغییرات خروجی در زمان رسیدن به لبه کلاک رخ میدهند و چنین glitch هایی وجود ندارند.

البته ماشین میلی نیز در موارد کوچکی مزیت هایی دارد. مثلا چون ماشین میلی تابع ورودی نیز هست ، دست طراح در طراحی را بازتر کرده و در شرایط مشابه نسبت به مور، میتواند تعداد استیت های کمتری را برای ساخت خروجی ایجاد کند. اما در کل به دلیلی که ذکر شد ، ماشین مور مرجح است.