به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

سیستم‌های دیجیتال 1

استاد: دکتر نوابی

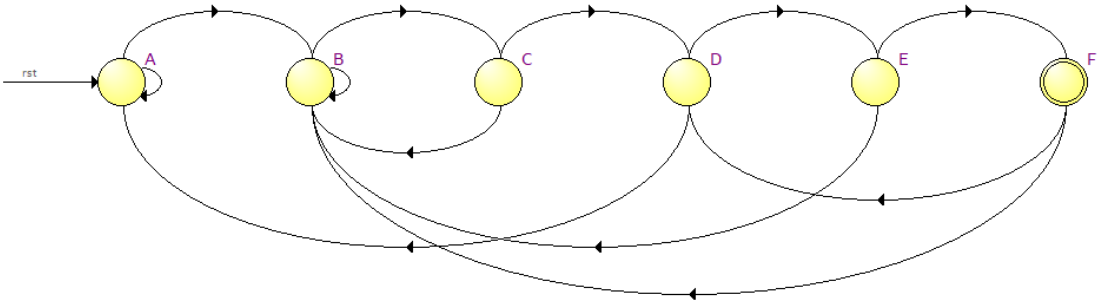
تمرین کامپیوتری شماره 5

فربد سیاه‌کلی

شماره دانشجویی: 810198510

فروردین 1400

1) دیاگرام detector عدد 10010 به شیوه moore machine، به صورت زیر است:

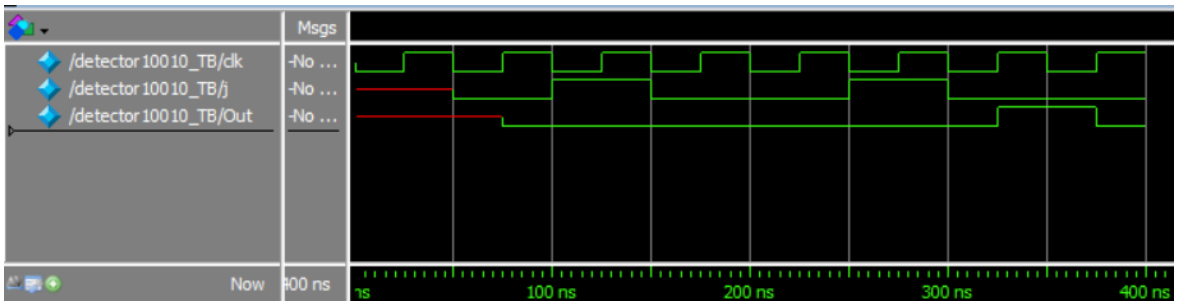


کد وریلاگ مربوطه:

```
`timescale 1ns/1ns
module detector10010_PRE(input clk, rst, J, output w);
  reg [2:0] ns, ps;
  parameter [2:0] A=3'b000, B=3'b001, C=3'b010, D=3'b011, E=3'b100, F=3'b101;
  always @(ps, J) begin
    ns = A;
    case(ps)
       A: ns = J ? B : A;
       B: ns = J ? B : C;
       C: ns = J ? B : D;
       D: ns = J ? E : A;
       E: ns = J ? B : F;
       F: ns = J ? B : D;
       default: ns = A;
     endcase
  end
  assign  w = (ps == F) ? 1'b1 : 1'b0;
  always @(posedge clk, posedge rst) begin
       if(rst)
           ps <= A;
       else
           ps <= ns;
  end
endmodule
```

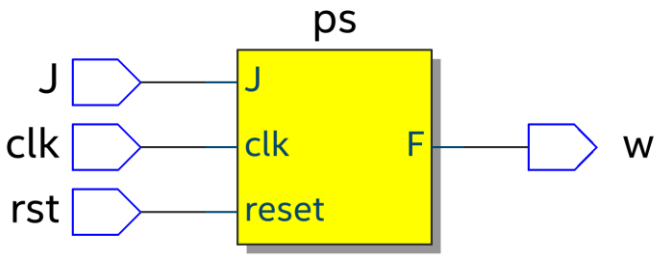
تست بنچ مربوطه:

خروجی کد که همانطور که مشاهده می‌شود عملیات detect در آن به درستی انجام شده است.

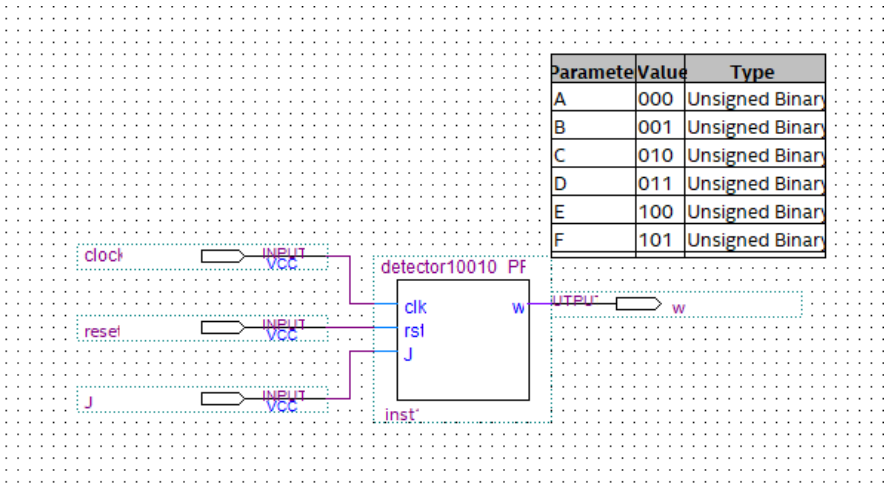


حال با استفاده از کوارتز، کد سنتز شده و خروجی‌های vo. و sdo. که شامل دیلی‌های ماژول است دریافت شده و در پایان کد اولیه و کد سنتز شده در کنار هم اجرا شده‌اند تا تفاوت قابل مشاهده باشد.

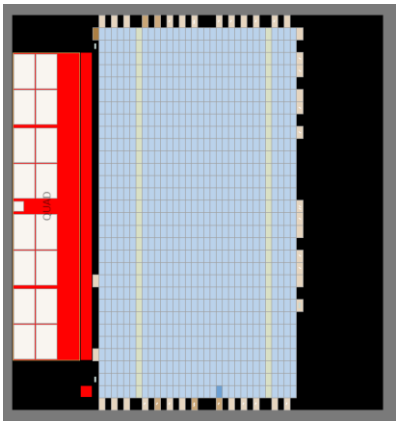
رسم RTL view ماژول:



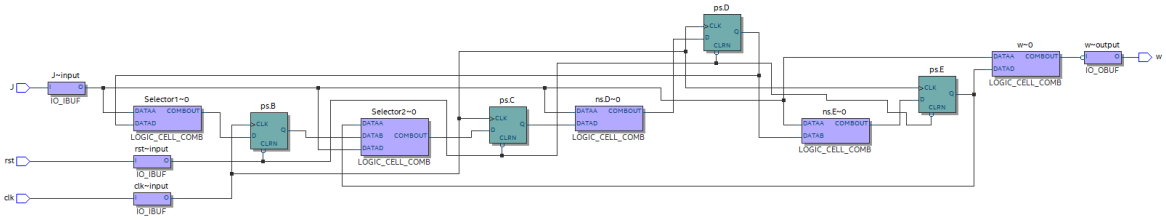
بلاک دیاگرام ماژول:



دریافت chip planner ماژول:



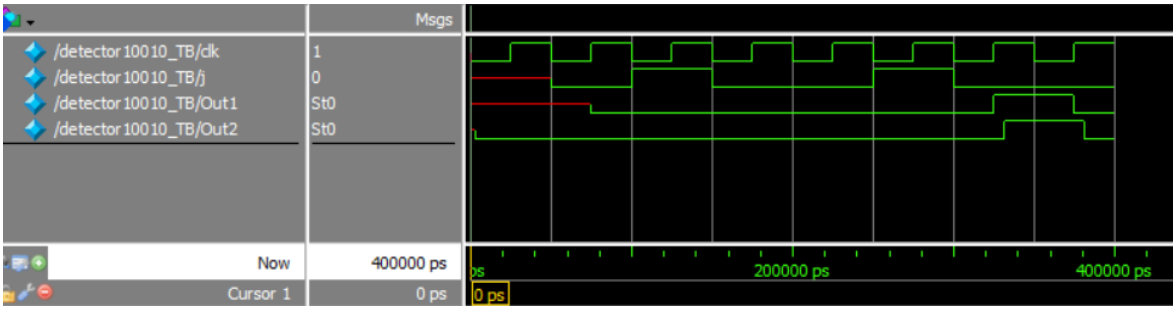
و در نهایت post mapping ماژول:



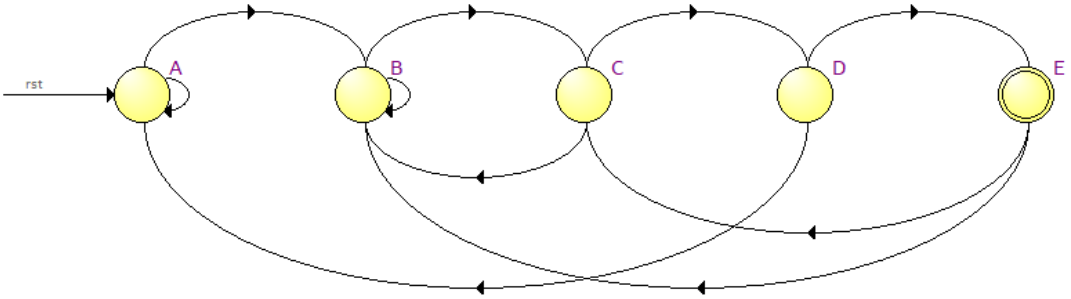
تست بنچ برای مقایسه و دیدن دیلی:

```
`timescale 1ns/1ns
module detector10010_TB();
   reg j, clk = 0, rst = 0;
   wire Out1, Out2;
   detector10010_PRE UUT1(clk, rst, j, Out1);
   detector10010 UUT2(clk, rst, j, Out2);
   always #25 clk <= ~clk;
   initial begin
      #50 j = 0;
      #50 j = 1;
      #50 j = 0;
      #50 j = 0;
      #50 j = 1;
      #50 j = 0;
      #100 $stop;
   end
endmodule
```

خروجی:



2)) دیاگرام detector عدد 10010 به شیوه mealy machine، به صورت زیر است:



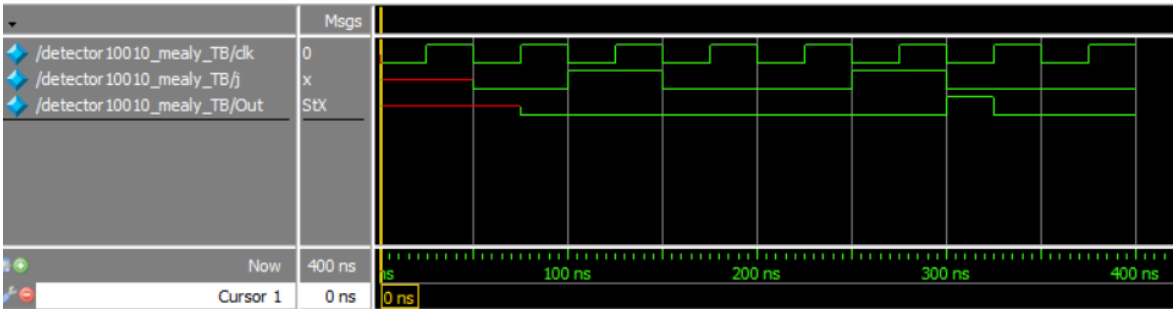
کد وریلاگ مربوطه:

```
`timescale 1ns/1ns
module detector10010mealy_PRE(input clk, rst, J, output w);
  reg [2:0] ns, ps;
  parameter [2:0] A=3'b000, B=3'b001, C=3'b010, D=3'b100, E=3'b011;
  always @(ps, J) begin
    ns = A;
    case(ps)
       A: ns = J ? B : A;
       B: ns = J ? B : C;
       C: ns = J ? B : D;
       D: ns = J ? E : A;
       E: ns = J ? B : C;
       default: ns = A;
     endcase
  end
  assign  w = (ps == E) ? J : 1'b0;
  always @(posedge clk, posedge rst) begin
       if(rst)
           ps <= A;
       else
           ps <= ns;
  end
endmodule
```

تست بنچ مربوطه:

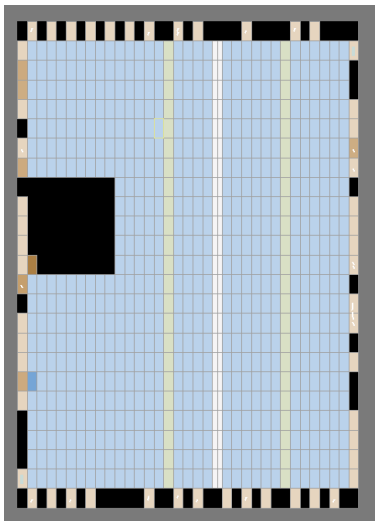
```
`timescale 1ns/1ns
module detector10010_TB();
   reg j, clk = 0, rst = 0;
   wire Out1, Out2;
   detector10010_PRE UUT1(clk, rst, j, Out1);
   detector10010 UUT2(clk, rst, j, Out2);
   always #25 clk <= ~clk;
   initial begin
      #50 j = 0;
      #50 j = 1;
      #50 j = 0;
      #50 j = 0;
      #50 j = 1;
      #50 j = 0;
      #100 $stop;
   end
endmodule
```

خروجی کد که همانطور که مشاهده می‌شود عملیات detect در آن به درستی انجام شده است.

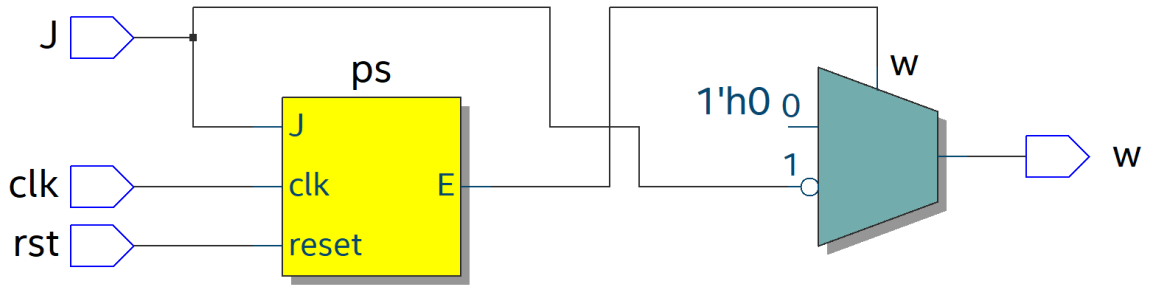


حال با استفاده از کوارتز، کد سنتز شده و خروجی‌های vo. و sdo. که شامل دیلی‌های ماژول است دریافت شده و در پایان کد اولیه و کد سنتز شده در کنار هم اجرا شده‌اند تا تفاوت قابل مشاهده باشد.

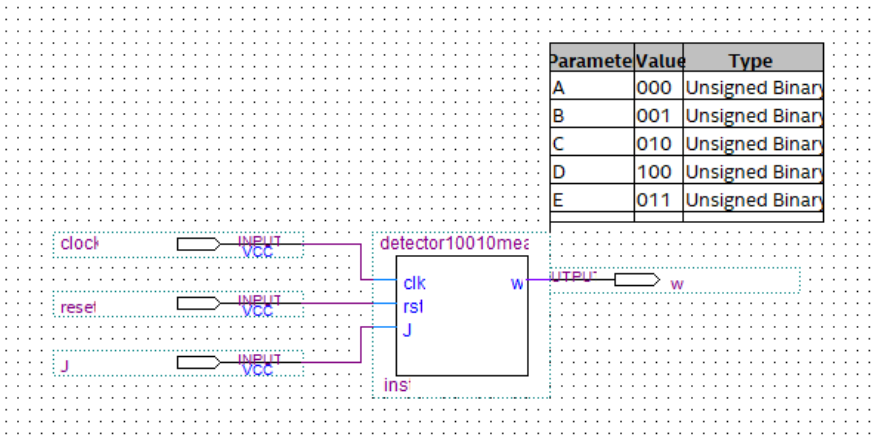
دریافت chip planner ماژول:



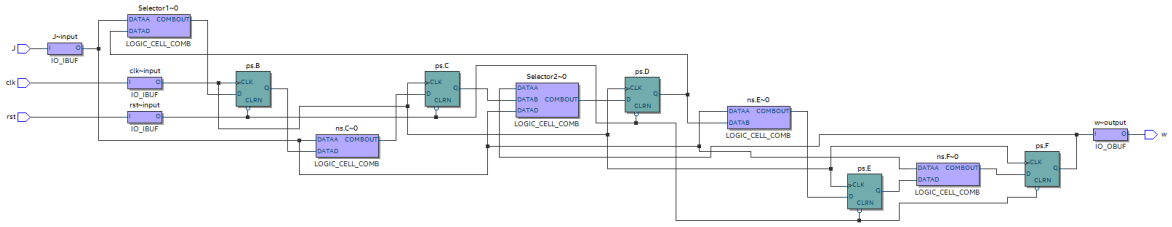
رسم RTL view ماژول:



بلاک دیاگرام ماژول:



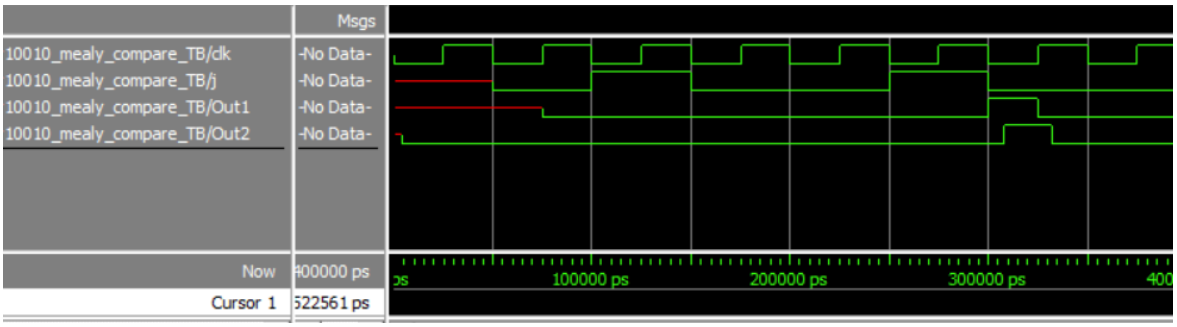
و در نهایت post mapping ماژول:



تست بنچ برای مقایسه و دیدن دیلی:

```
`timescale 1ns/1ns
module detector10010_mealy_TB();
   reg j, clk = 0, rst = 0;
   wire Out1, Out2;
   detector10010mealy_PRE UUT3(clk, rst, j, Out1);
   detector10010mealy UUT4(clk, rst, j, Out2);
   always #25 clk <= ~clk;
   initial begin
      #50 j = 0;
      #50 j = 1;
      #50 j = 0;
      #50 j = 0;
      #50 j = 1;
      #50 j = 0;
      #100 $stop;
   end
endmodule
```

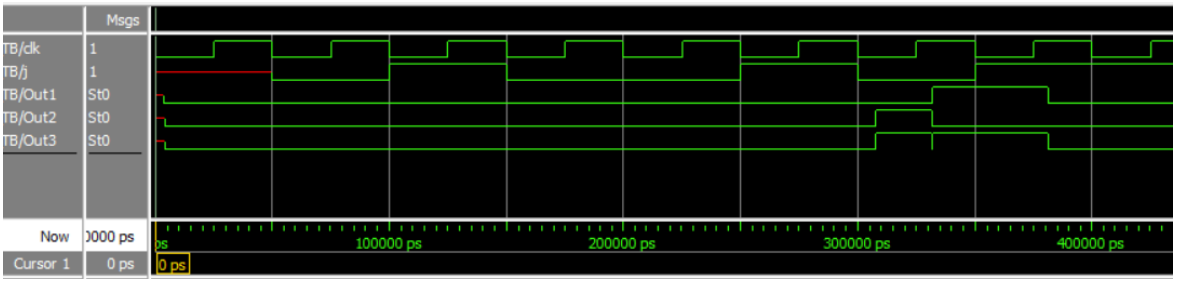
خروجی:



3) حال ماژول‌های دارای دیلی را در کنار هم ران کرده و همچنین در OUT3، XOR شده‌ی آنها را رسم می‌کنیم. تست بنچ مربوطه به شرح زیر است:

```
`timescale 1ns/1ns
module moore_and_mealy_TB();
   reg j, clk = 0, rst = 0;
   wire Out1, Out2, Out3;
   assign Out3 = Out1 ^ Out2;
   detector10010 UUT5(clk, rst, j, Out1);
   detector10010mealy UUT6(clk, rst, j, Out2);
   always #25 clk <= ~clk;
   initial begin
      #50 j = 0;
      #50 j = 1;
      #50 j = 0;
      #50 j = 0;
      #50 j = 1;
      #50 j = 0;
      #50 j = 1;
      #100 $stop;
   end
endmodule
```

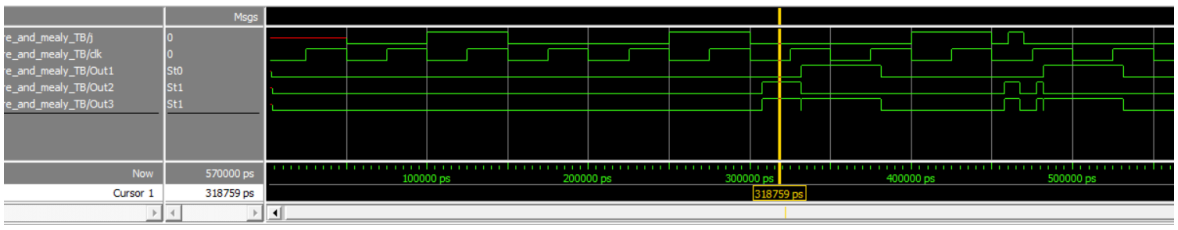
خروجی:



در میلی ماشین، خروجی به استیتی که در آن هستیم و همچنین ورودی ماشین بستگی دارد اما در مور ماشین خروجی تنها به استیت حال حاضر بستگی دارد.

در نتیجه میلی ماشین به صورت Asynchronous و مور ماشین به صورت Synchronous می‌باشند که به همین دلیل اگر ورودی دچار نوساناتی باشد، خروجی نیز تحت تاثیر آن قرار می‌گیرد.

تفاوت دو ماشین در شکل زیر قابل مشاهده است.



در انتها با استفاده از استیت زیر، گلیچ را برطرف می‌کنیم.

Out4 = Out3 | (Out1 & Out2)

همانطور که مشاهده می‌شود، گلیچ از بین رفت.