GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ

ELM 235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 0x4 Deney Raporu

Sonlu Otomatlar

Hazırlayanlar
1) 1801022024 – M. Cemal Eryiğit
2) 1801022077 – Burak Kamil Çiftci

Bu labın amacı

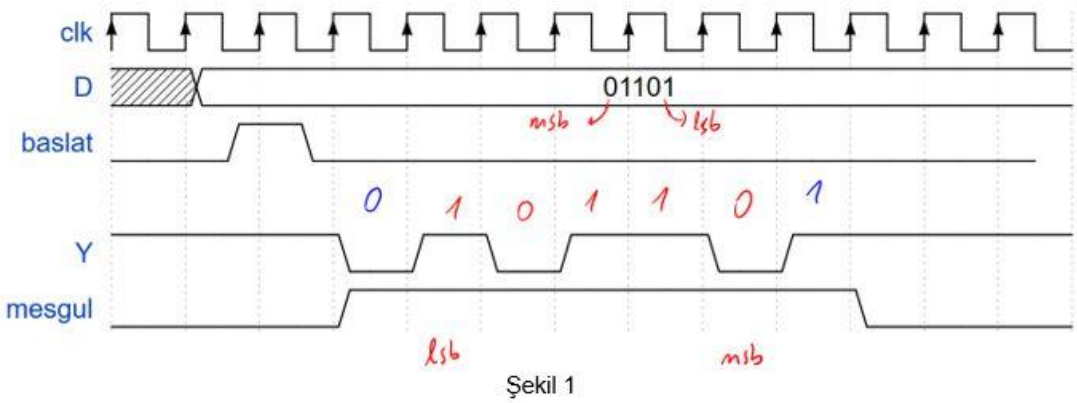
- Sonlu otomatlar oluşturabilmek.
- Tasarlanan sonlu otomatı DTD kullanarak gerçekleyip test edebilmek.

Problem 1 - Sıralarıyıcı

Bu problemde 5-**bit D** değerini **baslat** geldiğinde aşağıdaki isterlere uygun olarak **Y** çıkışından göndermeniz istenmektedir. (normal bir shift register olarak çalışacağını düşünebilirsiniz.)

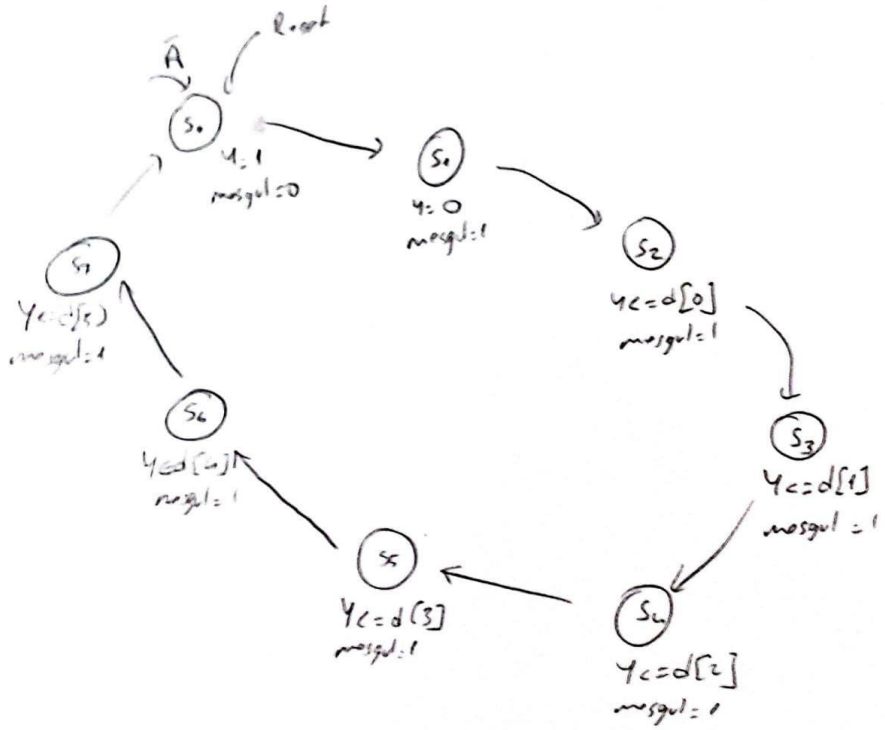
- Y çıkışı, baslat girişi 0 olduğu zaman lojik 1 olarak sürülecektir. FSM in S0 stateti burası olsun.
    - baslat girişi sadece 1 clock cycle süreyle aktif hale gelebilir.
- baslat 1 olduktan sonra önce 1 clock cycle lık lojik 0 gönderilecek
- sonrasında D nin LSB bit inden başlayarak her clock cycle da bir D nin bir sonraki biti gönderilecek.
- D nin bütün bitleri bittikten sonra 1 clock cycle lık lojik 1 gönderilecek
- FSM S0 statetine geri dönecek
- Devre S0 stateti haricindeki bütün state lerde **mesgul** çıkışı lojik 1 olarak sürülecek, S0 statetinde **mesgul** çıkışı lojik 0 olarak sürülecektir.
- İlk baştaki lojik 0 ı göndermek için bir state belirleyin.
- En sonraki lojik 1 i göndermek için bir state belirleyin.
- Aradaki D sinyallerini göndermek için state veya stateler belirleyin.

Örnek olarak D sinyalinin 01101 olduğunu varsayalım. start Y ve mesgul sinyallerinin zamanlama şeması Şekil 1 de verilmiştir.



Şekil 1

A. State transition diagramını çıkarınız.



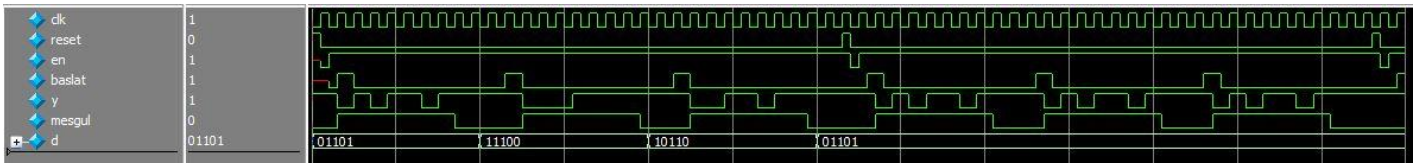
B. Devrenizi farklı kombinasyonlarla test ederek çalıştığını gözlemleyin.


```
module lab4_g24_p1(                                    //rtl
input logic clk, reset, en,A
input logic [4:0] D,
output logic Y,mesgul
);
typedef enum{s0,s1,s2,s3,s4,s5,s6,s7} statetype;
statetype state,nextstate;
always_ff @(posedge clk, posedge reset)
   begin
      if(reset) state<=s0;
      else if(en) state<=nextstate;
   end
```

```
always_comb

case(state)


s0:

begin

mesgul=0;

Y=1;

if(A) nextstate =s1;

else nextstate=s0;

end

s1:

begin

   mesgul=1;

   Y=0;

   nextstate = s2;

end

s2:

begin

   mesgul=1;

   Y<=D[0];

   nextstate =s3;

end

s3:

begin

   mesgul=1;

   Y<=D[1];

   nextstate=s4;

end
```

```
s4:
begin
   mesgul=1;
   Y<=D[2];
   nextstate=s5;
end
s5:
begin
   mesgul=1;
   Y<=D[3];
   nextstate =s6;
end
s6:
begin
   mesgul=1;
   Y<=D[4];
   nextstate =s7;
end
s7:
begin
   mesgul=1;
   Y=1;
   nextstate =s0;
end
   default:
      nextstate=s0;
   endcase
endmodule
```

```
timescale 1ns/1ps                                                    //testbench

module tb_lab4_g24_p1();

logic clk,reset,en,A,Y,mesgul;

logic [4:0]D;

lab4_g24_p1 dut0(.en(en),.clk(clk),.reset(reset),.Y(Y),.D(D),.A(A),.mesgul(mesgul));

always begin

clk=0; #10;

clk=1; #10;

end

always begin

reset=1; #10;

reset=0;

en=0; #10;

en=1;


A=0; #10;

A=1; #10; A=0; #90;

A=1; #10; A=0; #90;

A=1; #10; A=0; #90;

end

initial begin

d=5'b01101; #100;

d=5'b11100; #100;

d=5'b10110; #100;

d=5'b01101; #100;

#200;

$stop;

end

endmodule
```



Şekil 1 Dalga görüntüsü

Sonuç ve Yorum

Yaptığımız işlem sayesinde d girişinden alınan her bit icin belirlenen isterler y cıkışı icin oluşumu gözlemlendi. Bu sayede FSM kullanarak right-local shift register elde edildi. Bu sayede oluşturulan loop ile dongu durum kontrolu yapıp cıkış Y icin değer ataması yaptı.

Genel itibariyle pattern yakalayıcı devreyi istenilen sonucu oluşturacak şekilde tasarladık ve bize verilen görevi tamamladık. Bu sayede sonlu otommat tasarlama konusundaki uygulama becerimiz gelişti.