GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 0x4 Deney Raporu

Sonlu Otomatlar

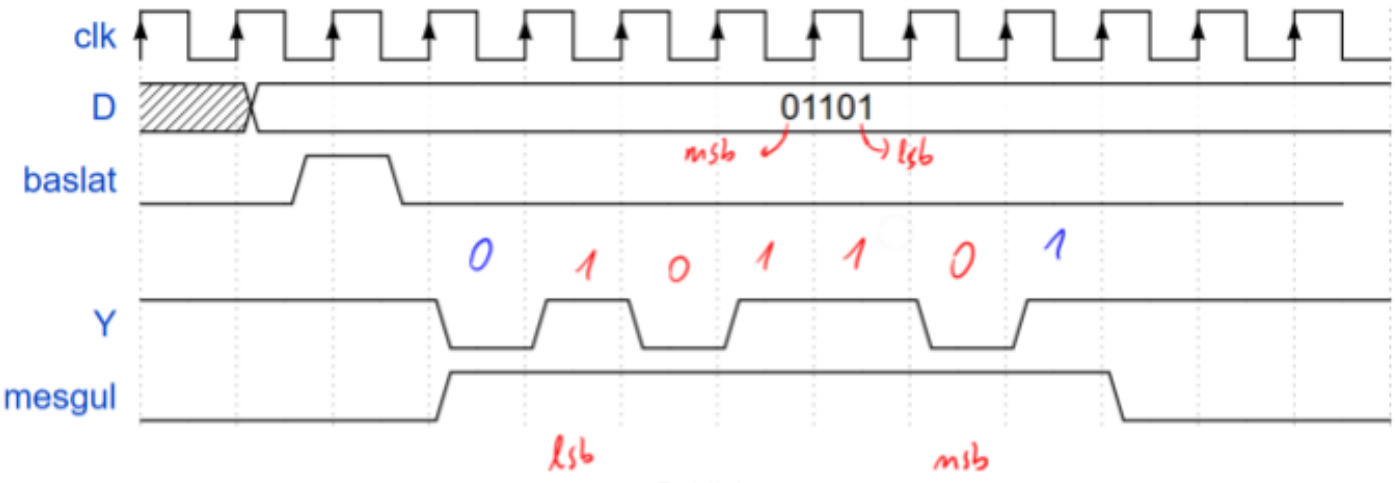
Hazırlayanlar
1) 1801022035 – Ruveyda Dilara Günal
2) 200102002087 – Alican Bayındır

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 meşgul sinyallerinin zamanlama şeması Şekil 1 de verilmiştir.



Şekil 1

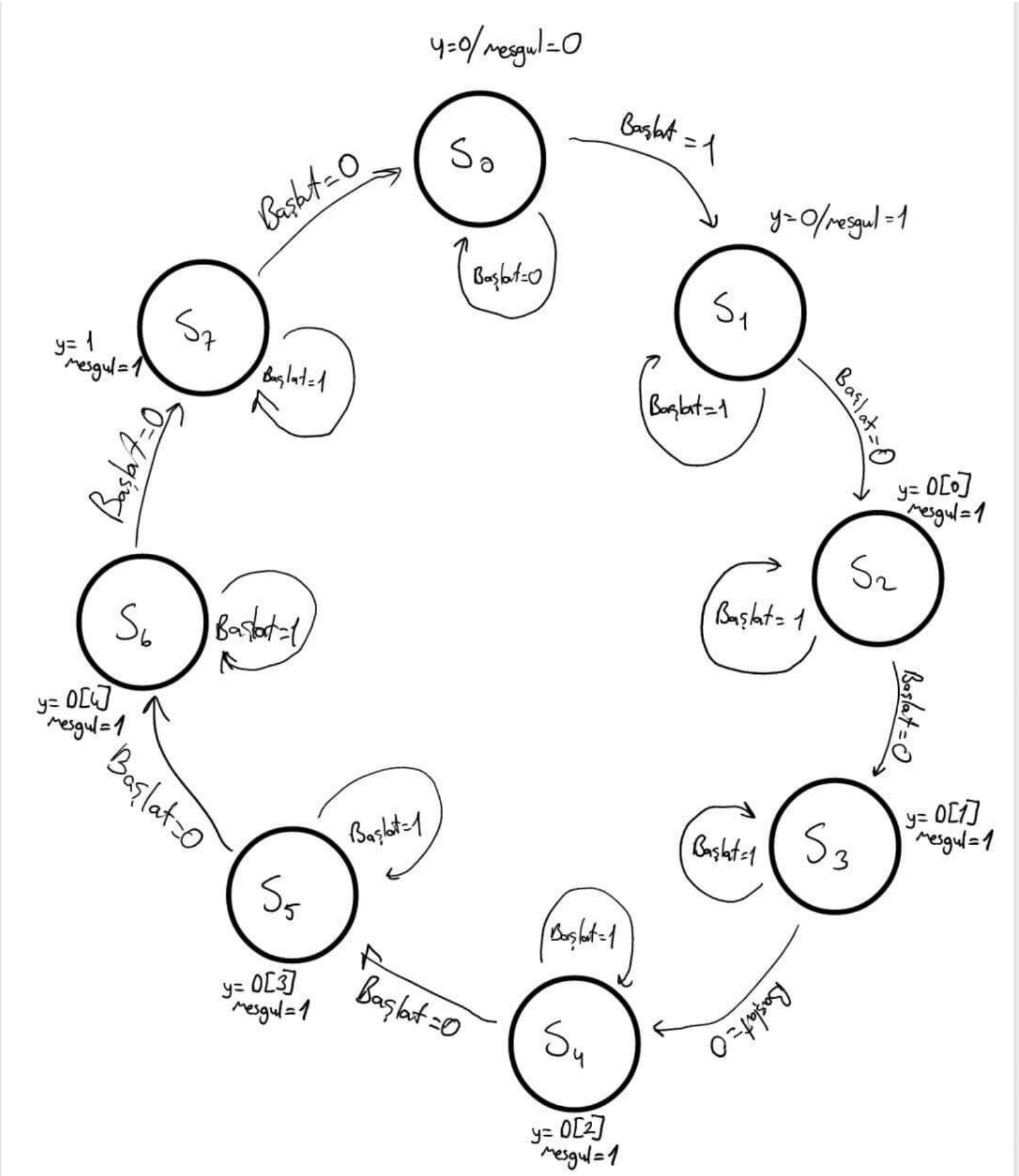
Not: D girişinin ne olduğunun önemi yoktur. Önemli olan gönderilen bitlerin sırasıdır.

```
module p1 (
input logic clk, reset, en,
input logic [4:0] D,
input logic baslat,
output logic Y,
output logic mesgul
);
```

A. State transition diagramını çıkarınız.

B. Devrenizi farklı kombinasyonlarla test ederek çalıştığını gözlemleyin.

A. Devrenin State transition diagramı.



Şekil 1 State transition diagram

B. Devreyi çalıştırmak için gerekli modülün kodları aşağıda görüldüğü gibidir.

```
module lab4_g29_p1 (
        input logic         clk, reset,
        input logic [4:0] D,
        input logic         baslat,
        output logic        y,
        output logic     mesgul );
typedef enum { s0,s1, s2, s3, s4, s5, s6, s7} statetype;
statetype state,next_state;


always_ff @(posedge clk)
begin
if(reset)
state <= s0;
else
state <= next_state;
end


always_comb
begin
case(state)
        s0:
        begin
         if(!baslat)
           next_state = s0;
            else
       next_state = s1;
```

```
    end
    s1:
     begin
    if(!baslat)
      next_state = s2;
       else
next_state = s0;
    end
    s2:
     begin
    if(!baslat)
      next_state = s3;
       else
next_state = s0;
    end
    s3:
     begin
    if(!baslat)
      next_state = s4;
       else
next_state = s0;
    end
    s4:
     begin
    if(!baslat)
      next_state = s5;
       else
```

```
        next_state = s0;
          end
          s5:
            begin
           if(!baslat)
             next_state = s6;
               else
        next_state = s0;
          end
          s6:
            begin
           if(!baslat)
             next_state = s7;
               else
        next_state = s0;
          end
          s7:
            begin
             next_state = s0;
          end
default: next_state = s0;
endcase
end

always_comb
begin
case(state)
```

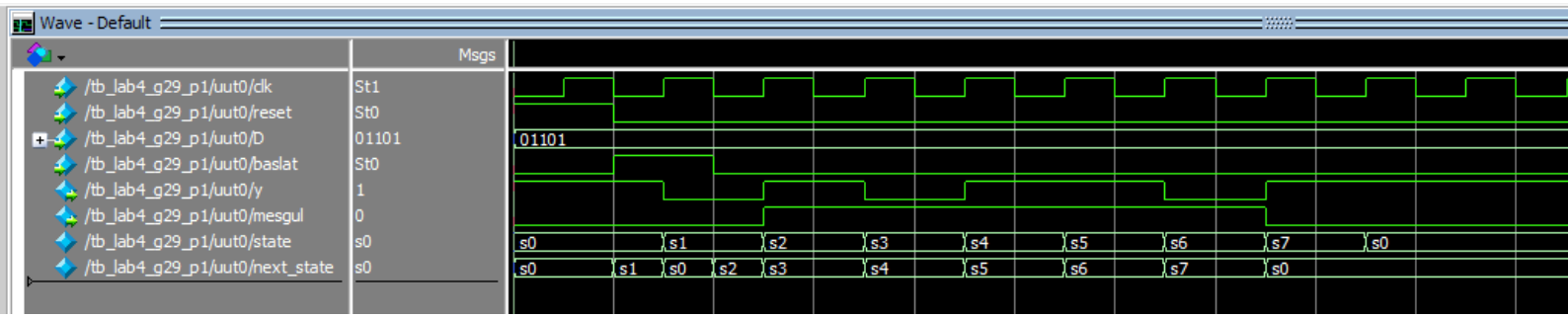
```
s0:
begin
mesgul = 0;
y = 1;
end
s1:
begin
mesgul = 0;
y = 0;
end
s2:
begin
mesgul = 1;
y = D[0];
end
s3:
begin
mesgul = 1;
y = D[1];
end
s4:
begin
mesgul = 1;
y = D[2];
end
s5:
begin
```

```
            mesgul = 1;
            y = D[3];
            end
            s6:
            begin
            mesgul = 1;
            y = D[4];
            end
            s7:
            begin
            mesgul = 0;
            y = 1;
            end
endcase
end
endmodule
```

Yukarıdaki yazan kodu çalıştırmak için gerekli olan testbench;

```
module tb_lab4_g29_p1 ();
logic clk, reset, y, mesgul, baslat;
logic [4:0] D;

lab4_g29_p1 uut0(.clk(clk), .reset(reset), .y(y), .mesgul(mesgul), .baslat(baslat), .D(D) );

initial begin
clk = 0;
forever #10 clk = ~clk ;
end
```

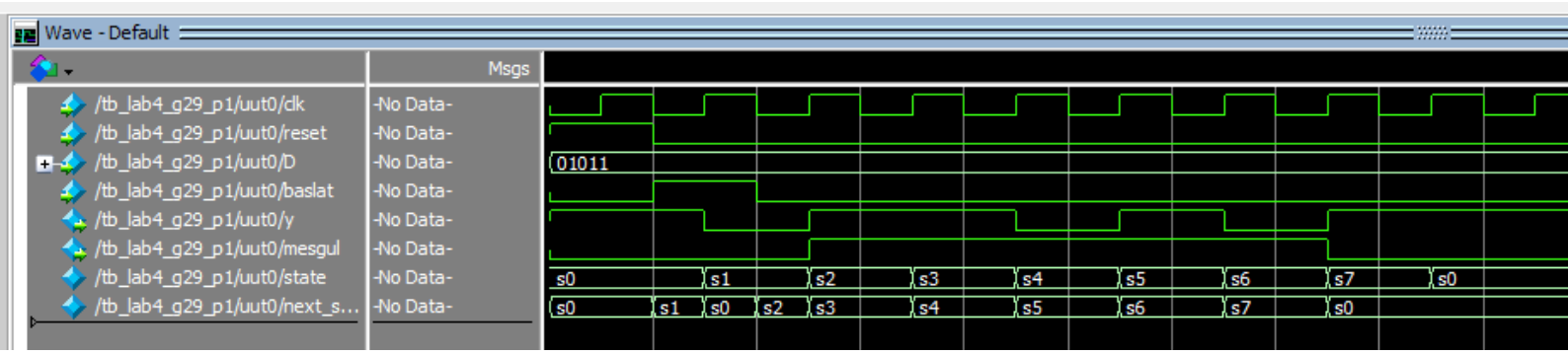
```
initial begin

reset = 1; baslat = 0; #20;

reset = 0; baslat = 1; #20;

baslat = 0; #150;

end


initial begin

D = 5'b01101 ;

end


initial begin

#400;

$stop;

end

endmodule
```



Şekil 2 Modelsim dalga formları.

Yukarıdaki simülasyon çıktısında da görüldüğü gibi başlat sinyali bir clock cycle süreyle bir olduktan sonra d giriş sinyali y çıkışına aktarılmıştır. Aktarılmaya başlamadan önce y çıkışına bir clock cyle süresi kadar 0, aktarıldıktan sonra ise bir clock cycle boyunca 1 sinyali verilmiştir.

Devre bir de d = 5’b11010 sinyali için denenmiştir. Ve başarılı bir sonuç elde edilmiştir

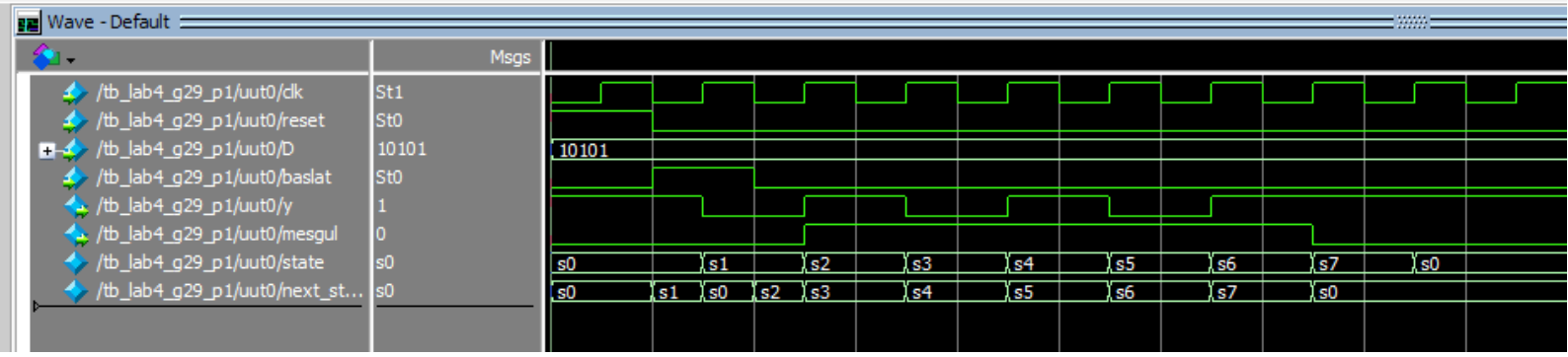


Şekil 3 Modelsim çıktı dalga formları 2.

Testbench üzerindeki initial begin bloğu aşağıdaki şekilde değiştirilmiştir.

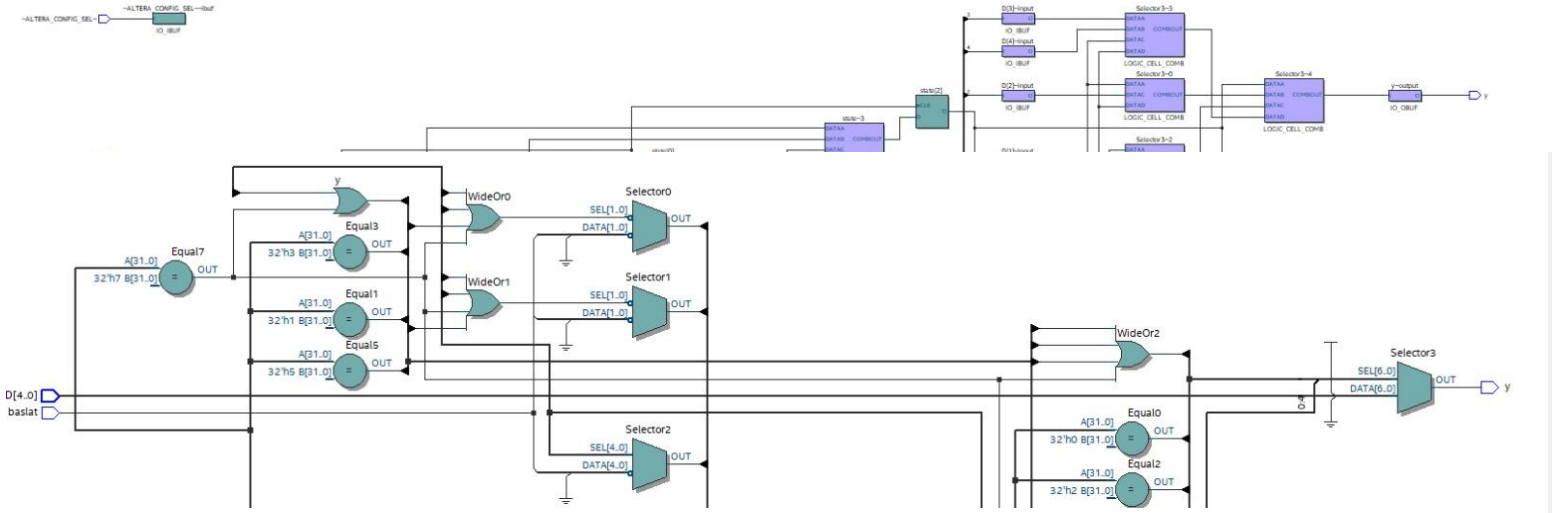
```
initial begin

D = 5'b10101; #50;

end
```

Bu değişiklik sonrası Şekil 3’teki gibi bir dalga formu gözlemlenmiştir.



Şekil 4 Modelsim dalga formları 3.

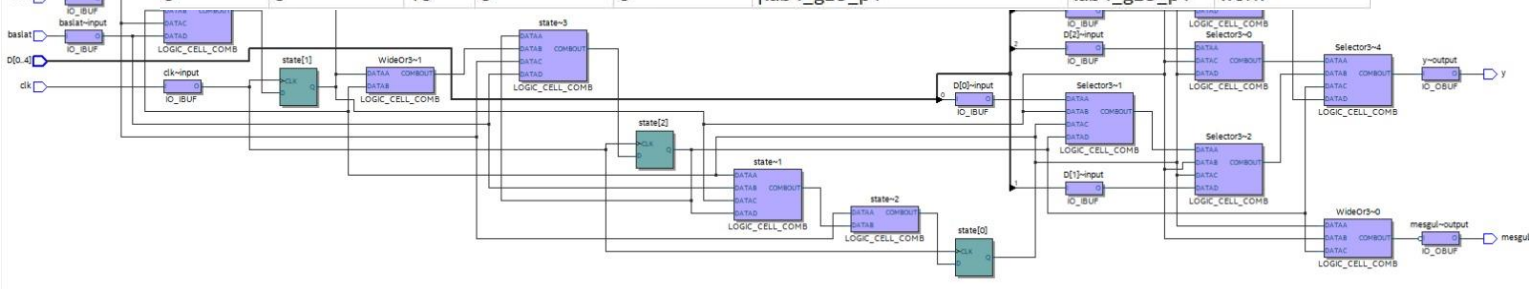
Devrenin Quartus çıktıları;



Şekil 7: Quartus Çıktısı - Post fitting

Şekil 5: Quartus Çıktısı -RTL

DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC blocks	Full Hierarchy Name	Entity Name	Library Name
0	0	10	0	0	|lab4_g29_p1	lab4_g29_p1	work



Şekil 6: Quartus Çıktısı - Post Mapping

Devrenin Clock transition döngüsü aşağıdaki gibidir. Döngüde de, daha önce belirtildiği gibi s4 state inde gene a=1 gelirse s4 state inde kalınır. Aynı durum s8 durumunda için de geçerlidir.s8 state inde iken bir kez daha ~a gelirse (a==0) state s8 olarak kalınır. Kod kısmında ilk olarak modüle içinde portlar tanımlanmıştır.Daha sonra enum içerisinde stateler tanımlanmıştır(s0,s2,s1 vb).Daha sonra always_ff bloğu içerisnde her clock transition da state'in durumunu belirleyecek nextstate state'e atanmıştır. (Her clock rising edge de).

	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Memory Bits	UFM Blocks	DSP Elements
1	lab4_g29_p1	11 (11)	3 (3)	0	0	0

Şekil 8: Resource Utilization Report

	Resource	Usage
1	Estimated Total logic elements	11
2		
3	Total combinational functions	11
4	⌄ Logic element usage by number of LUT inputs	
1	-- 4 input functions	7
2	-- 3 input functions	2
3	-- <=2 input functions	2
5		
6	⌄ Logic elements by mode	
1	-- normal mode	11
2	-- arithmetic mode	0
7		
8	⌄ Total registers	3
1	-- Dedicated logic registers	3
2	-- I/O registers	0
9		
10	I/O pins	10
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	state[1]
15	Maximum fan-out	8
16	Total fan-out	56
17	Average fan-out	1.65

Şekil 9: Resource Usage Summary

Devrede toplam 11 logic element ve 3 adet register kullanılmıştır. Devre bir yerde en fazla 8 dallanma olmuştur. Toplamda ise 56 dallanma oluşmuştur.

Sonuçlar ve Genel Yorumlar;

Bu laboratuvar föyünce verilen sıralayıcı devresi istenen bütün isterleri yerine getirebilecek ölçüde başarılı bir biçimde tasarlanmıştır. Tasarlanan sıralayıcı devresi başarılı bir biçimde D sinyali olarak verilen değeri çıkışta sıralamaktadır. Bu föyde temel olarak FSM devrelerinin kağıt üzerinde tasarlanması ve system verilog kullanılarak nasıl tasarlanabileceği öğrenilmiştir.

Referanslar:

[1] https://www.chipverify.com/verilog/verilog-n-bit-shift-register

[2] https://www.allaboutcircuits.com/technical-articles/understanding-verilog-shift-registers/

[3] Harris, S. and Harris, D., 2015. Digital Design and Computer Architecture: ARM Edition. Morgan Kaufmann.

[4] Quartus Prime Introduction Using Schematic Designs. URL: ftp://ftp.intel.com/Pub/fpgaup/pub/Intel_Material/16.0/Tutorials/Schematic/Quartus_II_Introduction.pdf Accessed: 21.02.2020

[5] https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/msgs/msgs/evrfx_veri_procedural_assignment.html