



GEBZE TEKNİK ÜNİVERSİTESİ  
ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 0x6 Deney Raporu

Sonlu Otomatlar

Hazırlayanlar
1) 1801022022 – Alperen Karataş
2) 1801022091 – Ogün Uygur Yıldırım

## 1. Giriş

Bu deney kapsamında sonlu otomatlar devreleri hakkında çalışılmıştır. Gerekli araştırmalar yapıp deneye başlanmıştır.

## 2. Problemler

### 2.1. Problem I – Pattern yakalayıcı

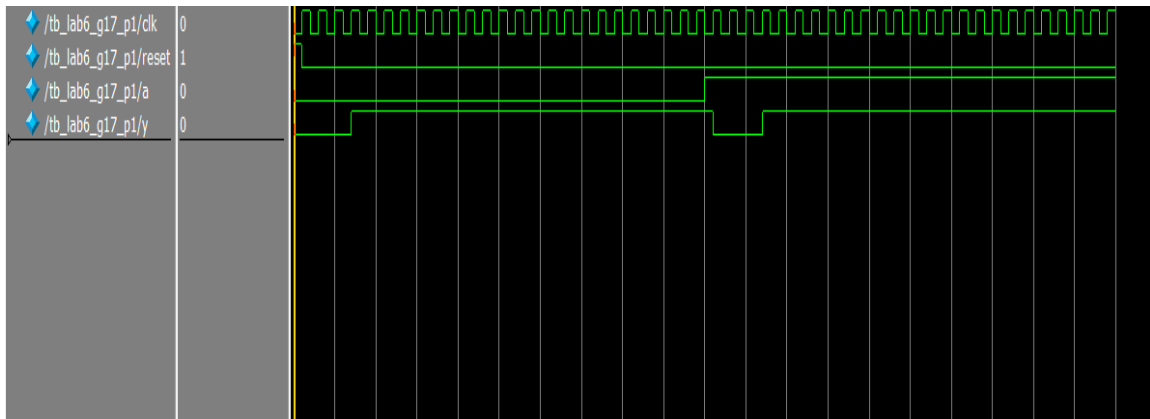
#### 2.1.1. Teorik Araştırma

Deneye başlanılmadan önce FSM devreleri hakkında bilgi sahibi olunup **systemverilog** dilinde nasıl uyarlanacağı araştırılmıştır.

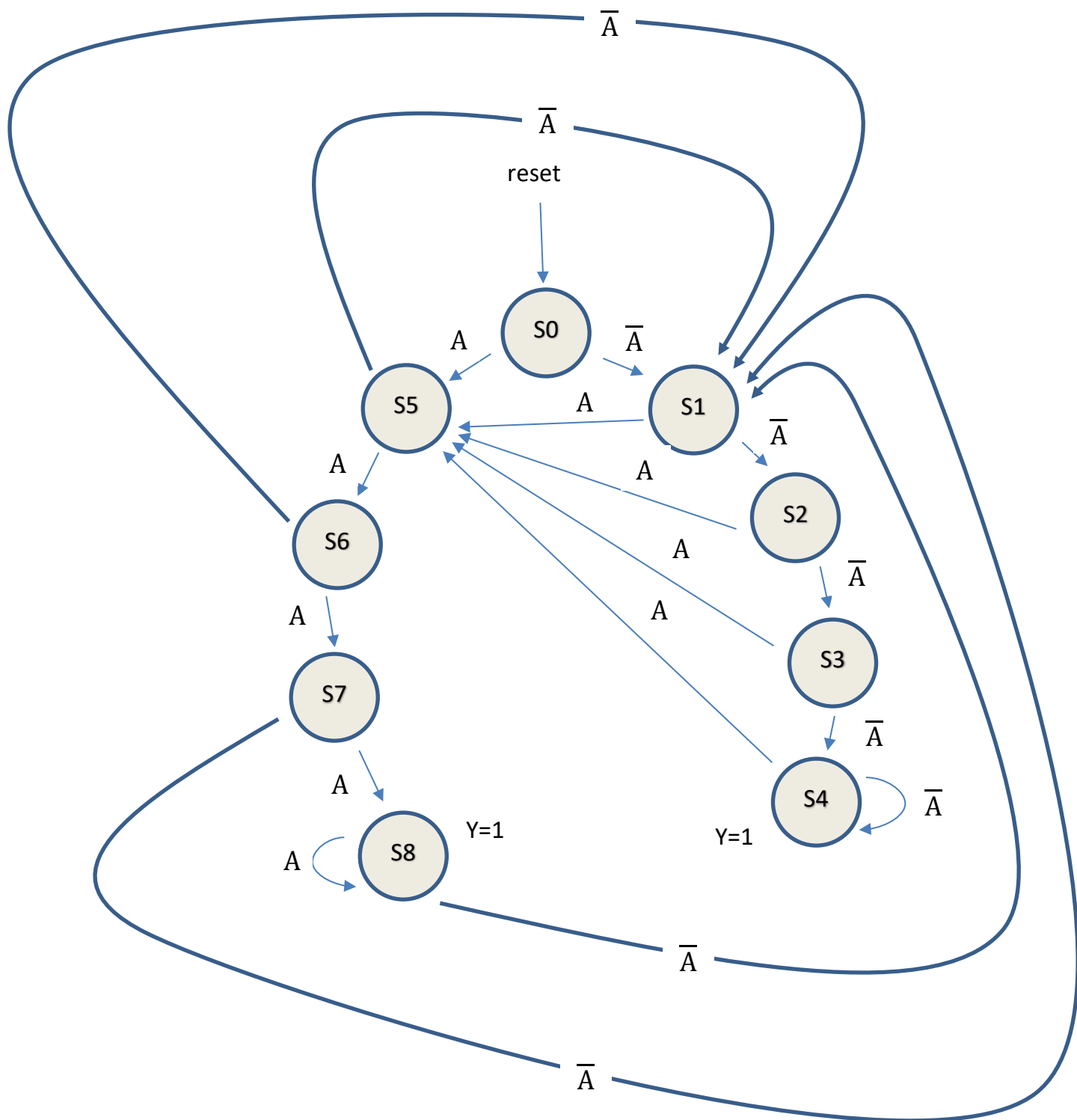
#### 2.1.2. Deneyin Yapılışı

Deneye başlamadan önce State Transition Diagram çıkarılmıştır ve bu diagrama göre gerekli kodlar yazılmıştır. Devre, giriş sinyalinden 4 kere ardarda 0 veya 4 kere ardarda 1 geldiği zaman sonucu verecek şekilde tasarlanmıştır.

Bir sonraki sayfada state transition diagram, Şekil 1’de ise devrenin simülasyon sonucu görülmektedir.



Şekil 1. Simülasyon çıktısı



	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Memory Bits	UFM Blocks	DSP Elements
1	lab6_g17_p1	9 (9)	4 (4)	0	0	0

**Şekil 2.** Utilization Report (1)

DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC blocks	Full Hierarchy Name	Entity Name	Library Name
0	0	0	4	0	0	lab6_g17_p1	lab6_g17_p1	work

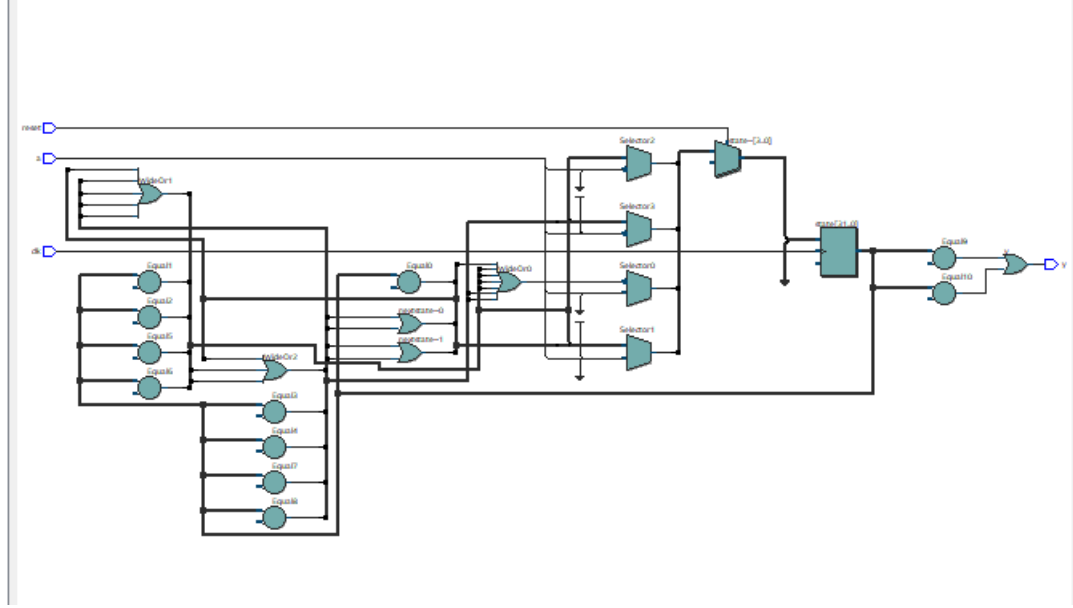
**Şekil 3.** Utilization Report (2)

Şekil 2 ve Şekil 3’de görüldüğü üzere devrede 9 adet “Combinational ALUT”, 4 adet lojik register saptanmıştır.

	Fmax	Restricted Fmax	Clock Name	Note
1	553.71 MHz	250.0 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

**Şekil 4.** Fmax değeri

Şekil 4’ten anlaşılacağı üzere devrenin maksimum çalışma frekansının 553.71 MHz olduğu görülmektedir.



**Şekil 5.** RTL şeması

Şekil 5’te devrenin RTL şeması görülmektedir.

### 2.1.3. Sonuçların Yorumu

Bu deney, problemde istenilen şekilde tasarlanmıştır ve şekil 1’de görüldüğü gibi istenilen sonuçlar elde edilmiştir. Simülasyon sonucu daha detaylı incelenirse 4 kere ardarda 0 veya 4 kere ardarda 1 olduğu örüntülerde, y çıkış değeri 1 olarak görülmüştür. 4’ten fazla tekrar olması durumunda y çıkış değerinin 1 olarak devam ettiği anlaşılmıştır. Çizilen state transition diagramına uygun olarak yazılan kodların, simülasyon ekranında da bizi istediğimiz sonuca ulaştırdığını görebiliyoruz.

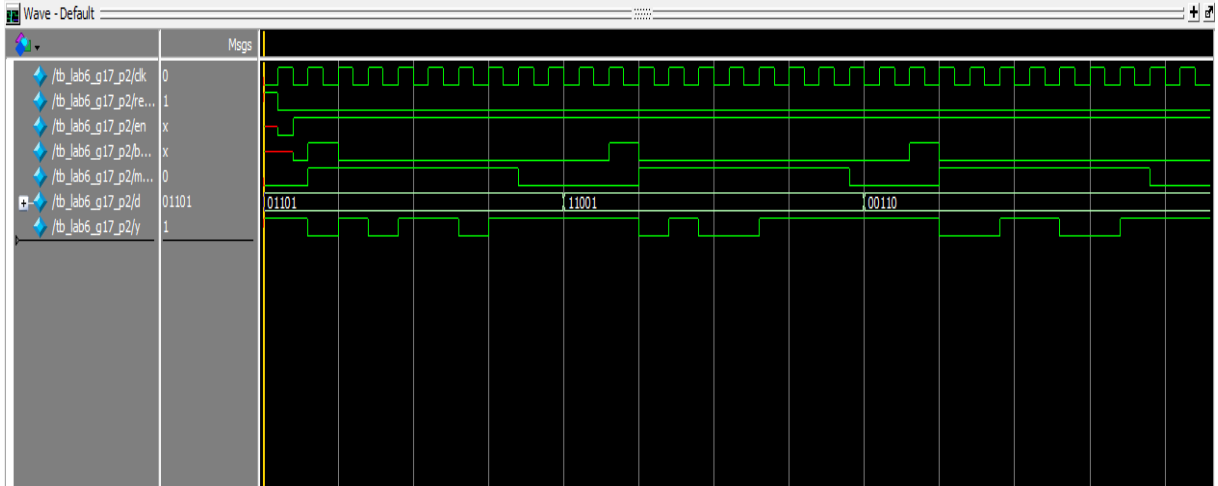
## 2.2. Problem II - Sıralayıcı

### 2.2.1. Teorik Araştırma

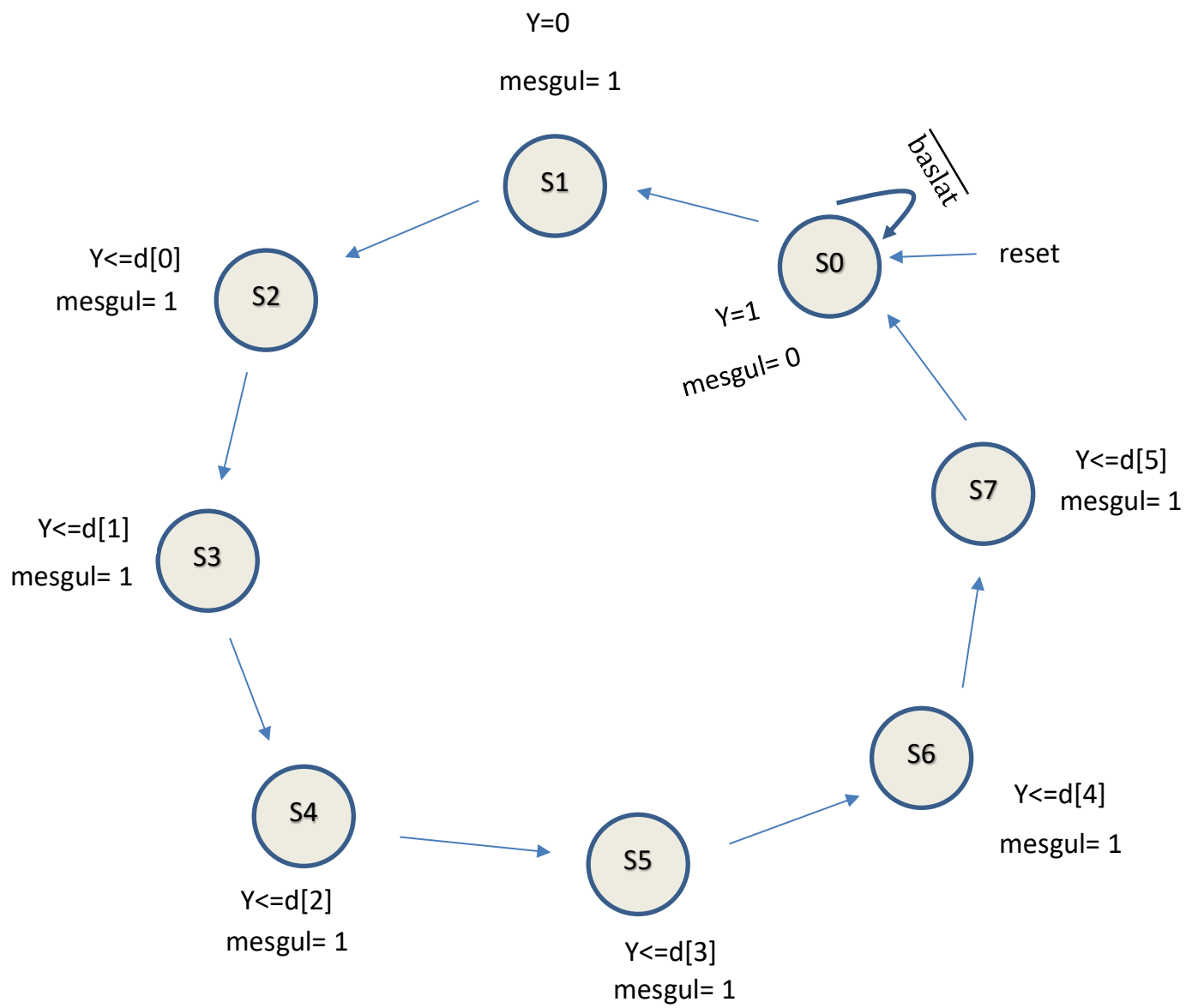
Deneye başlamadan önce sıralayıcı devreler hakkında bilgi sahibi olunmuştur. Devrenin gereken State Transition Diagramı hakkında araştırmalar yapılmıştır.

### 2.2.2. Deneyin Yapılışı

Bu deneyde öncelikle State Transition Diagramı çıkarılmıştır ve bu diagrama uygun olarak devre tasarımı yapılmıştır. Devrede 5 bit değerindeki başlat sinyali geldiğinde problemde belirtilen kurallara uygun olarak Y sinyal çıkışı gözlemlenmiştir. Devre farklı kombinasyonlarla test edilerek kontrol edilmiştir. Bir sonraki sayfada state transition diagram, Şekil 6’da ise devrenin simülasyon sonucu görülmektedir.



Şekil 6. Simülasyon çıktısı




	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Memory Bits	UFM Blocks	DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC bloc
1	lab6_g17_p2	10 (10)	3 (3)	0	0	0	0	0	11	0	0

**Şekil 7.** Utilization Report (1)

ADC blocks	Full Hierarchy Name	Entity Name	Library Name
0	lab6_g17_p2	lab6_g17_p2	work

**Şekil 8.** Utilization Report (2)

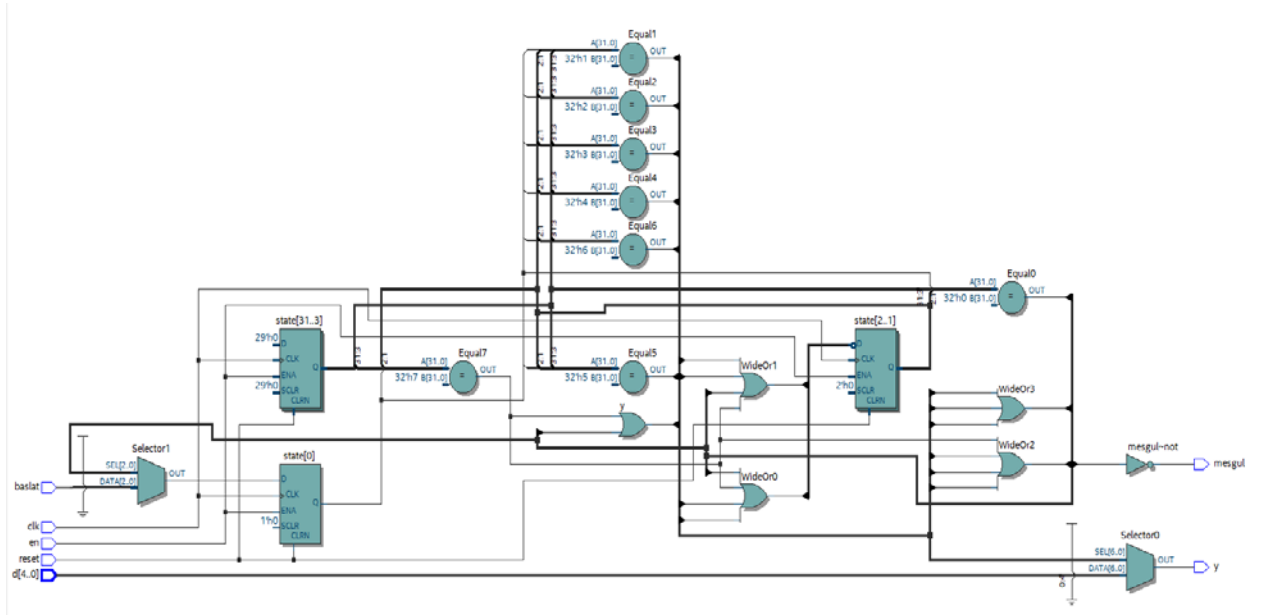
Şekil 7 ve Şekil 8’de görüldüğü üzere devrede 10 adet “Combinational ALUT”, 3 adet lojik register ve 11 adet pin görülmüştür.

Slow 1200mV 85C Model Fmax Summary				
 <<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	599.52 MHz	250.0 MHz	clk	lim...te)

**Şekil 9.** Fmax değeri

Şekil 9’ten anlaşılacağı üzere devrenin maksimum çalışma frekansının 599.52 MHz olduğu görülmektedir.





Şekil 10. RTL şeması

Şekil 10’da sıralayıcı devresinin RTL şeması görülmektedir.

### 2.2.3. Sonuçların Yorumu

Şekil 6’da anlaşıldığı üzere, testbench dosyasında girilen d giriş sinyal bit değerleri ile simülasyon çıktısında olan y çıkış sinyalinin eşit olduğu görüntülenmiştir. Böylelikle devrenin doğru çalıştığı anlaşılmıştır. Devre birden fazla kombinasyonlarla denenmiş ve testbenchte verilen her pattern çıkış sinyalinde gözlemlenmiştir. Bir önceki problemde de kazanılan tecrübelerle orantılı olarak simülasyon çıktılarının çıkarılan state transition diagrama uygunluğu bu problemde de kanıtlanmış oldu.

### 3. Sonular ve Genel Yorumlar

Bu laboratuvar alıřmasında, istenilen bir sorunun özümüne ulaşmak için öncelikle state transition diagramların nasıl ıkarılacağı hakkında bilgi sahibi olup, sonrasında bunun koda aktarımı tam anlamıyla anlaşılmıştır.

Problemlerde bizden istenilenler, adım adım anlaşıp sırasıyla state transition diagramına aktarıldı. Ařama aşama sorunu tahlil ederek ilerlemek, problemlerde başarıya ulaşmamızda en büyük etkendi. Yazılan kodların simölasyon ekranında ıktısı, bize state transition diagramın ve bizden sorunu istenen özümün doėruluėu veya yanlışlıėı hakkında fikir verdi. Kendi açımızdan düşündüğümüzde, problemin state transition diagramını ıkarmanın kodunu yazmaktan daha zorlayıcı ve düşündürücü olduėu yorumunu yapabiliriz.

Genel itibariyle pattern yakalayıcı devresini tasarlarken edindiğimiz tecrübe ve kazanımlar, sıralayıcı devresinin tasarımını yapmakta bize ışık tutmuş oldu. Bu sayede gerek diagram gerekse kodlamada karşılařtığımız sorunların üstesinden ok zorlanmadan gelebildik.

### 4. Referanslar

[1] Ders slaytları

[2] Harris and D.Harris, Digital Design and Computer Architecture: ARM Edition, 1st edition. Morgan Kaufmann, 2015

**\*/Bir sonraki sayfadan itibaren kodlar başlamaktadır/\***

## **KODLAR**

```
/*lab6_g17_p1.sv

*Hazırlayanlar:

* Alperen Karataş - Ogün Uygur Yıldırım

* ELM235 2020 Bahar Lab6 - Problem 1

* Pattern sayıcı

*/

module lab6_g17_p1(
input logic clk,reset,a,
output logic y
);
typedef enum{s0,s1,s2,s3,s4,s5,s6,s7,s8} statetype;
statetype state,nextstate;

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

always_comb
case(state)

s0:
begin
if(~a) nextstate =s1;
else nextstate =s5;
end

s1:
begin
if(~a) nextstate = s2;
else nextstate = s5;
end

s2:
begin
if(~a) nextstate =s3;
else nextstate =s5;
end
```

```

s3:
begin
if(~a) nextstate=s4;
else nextstate=s5;
end

s4:
begin
if(~a) nextstate=s4;
else nextstate=s5;
end

s5:
begin
if(~a) nextstate =s1;
else nextstate =s6;
end

s6:
begin
if(~a) nextstate =s1;
else nextstate =s7;
end

s7:
begin
if(~a) nextstate =s1;
else nextstate =s8;
end

s8:
begin
if(~a) nextstate =s1;
else nextstate=s8;
end
default nextstate=s0;
endcase

assign y=(state==s8 || state==s4)?1:0;

/*always_comb
y=0;
if(state==s8||state==s4)
begin
y=1;
end */

endmodule

```

```
/*tb_lab6_g17_p1.sv

*Hazırlayanlar:

* Alperen Karataş - Ogün Uygur Yıldırım

* ELM235 2020 Bahar Lab6 - Problem 1

* Pattern sayıcı testbench

*/

`timescale 1ns/1ps

module tb_lab6_g17_p1();
logic clk,reset,a,y;

lab6_g17_p1 uut0(.clk(clk),.reset(reset),.a(a),.y(y));

always begin
clk=0; #10;
clk=1; #10;
end

initial begin
reset=1; #10;
reset=0;
end

initial begin
a=0; #100;
a=0; #100;
a=0; #100;
a=0; #100;
a=0; #100;

a=1; #100;
a=1; #100;
a=1; #100;
a=1; #100;
a=1; #100;

$stop;
end

endmodule
```

```

/*lab6_g17_p2.sv

*Hazırlayanlar:

* Alperen Karataş - Ogün Uygur Yıldırım

* ELM235 2020 Bahar Lab6 - Problem 2

* Sıralayıcı

*/

module lab6_g17_p2(
input logic clk, reset, en,
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,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(baslat) 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
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
```

```

/*tb_lab6_g17_p2.sv

*Hazırlayanlar:

* Alperen Karataş - Ogün Uygur Yıldırım

* ELM235 2020 Bahar Lab6 - Problem 2

* Sıralayıcı testbench

*/

`timescale 1ns/1ps

module tb_lab6_g17_p2();
logic clk,reset,en,baslat,y,mesgul;
logic [4:0]d;

lab6_g17_p2
uut0(.en(en),.clk(clk),.reset(reset),.y(y),.d(d),.baslat(baslat),.mesgul
(mesgul));

always begin
clk=0; #10;
clk=1; #10;
end

always begin
reset=1; #10;
reset=0;
en=0; #10;
en=1;
baslat=0; #10;
baslat=1; #20;
baslat=0; #180;
baslat=1; #20;
baslat=0; #180;
baslat=1; #20;
baslat=0; #180;
end

initial begin
d=5'b01101; #200;
d=5'b11001; #200;
d=5'b00110;
#2000;
$stop;
end

endmodule

```