GEBZE TEKNİK

ÜNİVERSİTESİ

ELEKTRONİK

MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE

TASARLABORATUVARI

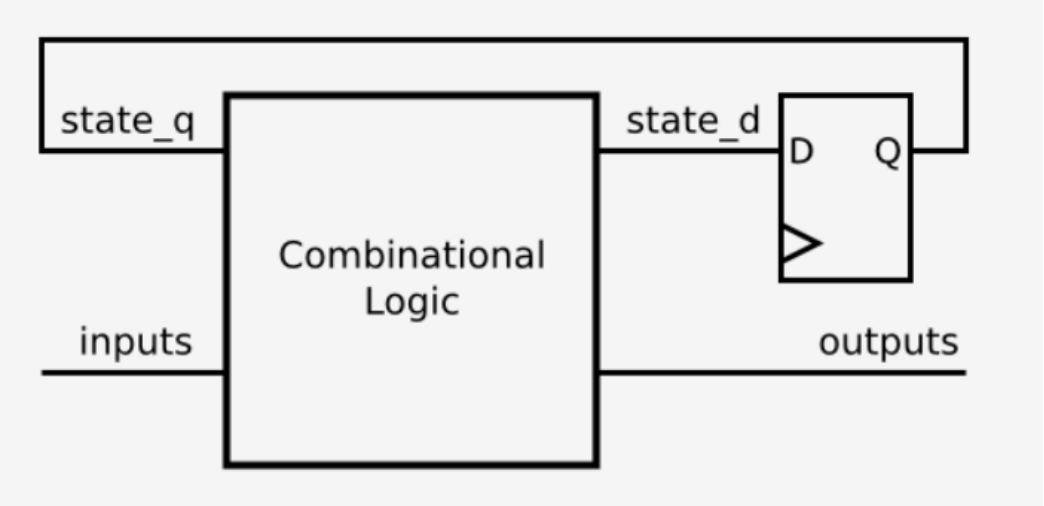
LAB 6 Deney Raporu

Sonlu Otomatlar

Hazırlayanlar
1) 200102002031 – Beyza Duran
2) 200102002043 – Senanur Ağaç

1. Teorik Araştırma

Bir FSM, en genel biçiminde, yalnızca mevcut durumu tutan bir iki duraklılık kümesi ve mevcut durum ve bazı girdiler verilen bir sonraki durumu belirleyen bir kombinasyonel mantık bloğudur. Çıktı, mevcut durumun ne olduğuna göre belirlenir.



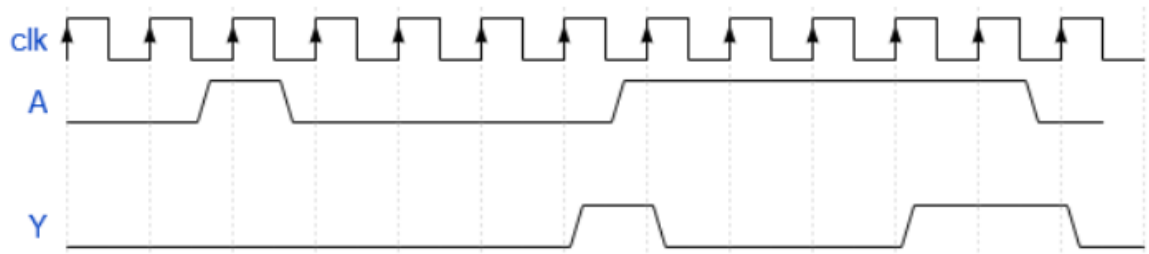
Bir sonlu-durum makinesi (FSM) veya basitçe bir durum makinesi , matematiksel bir hesaplama modelidir . Herhangi bir zamanda tam olarak sonlu sayıdaki durumlardan birinde olabilen soyut bir makinedir. FSM, bazı girdilere yanıt olarak bir durumdan diğerine geçebilir ; bir durumdan diğerine geçişe geçiş denir .]Bir FSM, durumlarının, ilk durumunun ve her geçişi tetikleyen girdilerin bir listesiyle tanımlanır. Sonlu durum makineleri iki tiptir - deterministik sonlu durum makineleri ve deterministik olmayan sonlu durum makineleri . Bir deterministik sonlu durum makinesi, deterministik olmayan herhangi bir makineye eşdeğer olarak oluşturulabilir.

Durum makinelerinin davranışı, sunuldukları olaylar dizisine bağlı olarak önceden belirlenmiş bir dizi eylem gerçekleştiren modern toplumdaki birçok aygıtta gözlemlenebilir. Basit örnekler , uygun madeni para kombinasyonu yatırıldığında ürün dağıtan otomatlar , durak sırası binicilerin talep ettiği katlara göre belirlenen asansörler , arabalar beklerken sırayı değiştiren trafik ışıkları ve gerekli olan şifreli kilitlerdir . uygun sırayla bir dizi sayının girişi.

1. Problemler

1.1 Problem 1 - Pattern yakalayıcı

Bir girişi (A), bir çıkışı olan bir devrede, giriş sinyalinden 4 kere ardarda 0 veya 4 kere ardarda 1 geldiğinde çıkışı (Y) 1 yapan bir FSM devresi tasarlayın. Eğer 4 den daha fazla tekrar varsa, çıkış 1 kalmaya devam etmesi gerekmektedir. Şekil 1 de zamanlama örneği verilmiştir.



Şekil 1

A. State Transition Diagramınızı çıkarın

B. Bu statelerinize göre devrenizi SystemVerilog da tasarlayıp, test edin

2.1.1 Problemin Çözümü

```
/* Hazırlayanlar: Senanur Ağaç – Beyza Duran
Lab6 problem 1 Pattern Yakalayıcı
*/
module lab6_g15_p1 (
input logic clk,reset,a,
output logic tick );
typedef enum {s0,s1,s2,s3,s4,s5,s6,s7,s8}
statetype ;
statetype state,nextstate ;// enum
always_ff @(posedge clk) // clock transition
        if(reset) state<=s0;
        else state<=nextstate ;
always_comb
case(state)
s0:begin
if(a) nextstate=s1;
else nextstate=s5; end
s1: begin
if(a) nextstate=s2;
else nextstate=s0; end
s2: begin
if(a) nextstate=s3;
else nextstate=s0; end
s3: begin
if(a) nextstate=s4;
else nextstate=s0; end
s4: begin
if(a) nextstate=s4;
else nextstate=s0; end
s5: begin
if(!a) nextstate=s6;
else nextstate=s0; end
s6: begin
if(!a) nextstate=s7;
else nextstate=s0; end
s7: begin
if(!a) nextstate=s8;
else nextstate=s0; end
s8: begin
if(!a) nextstate=s8;
else nextstate=s0; end
default: nextstate=s0;
endcase
always_comb
        if(state==s4)
                tick=1;
        else if(state==s8)
                tick=1;
        else
                tick=0;
endmodule
```

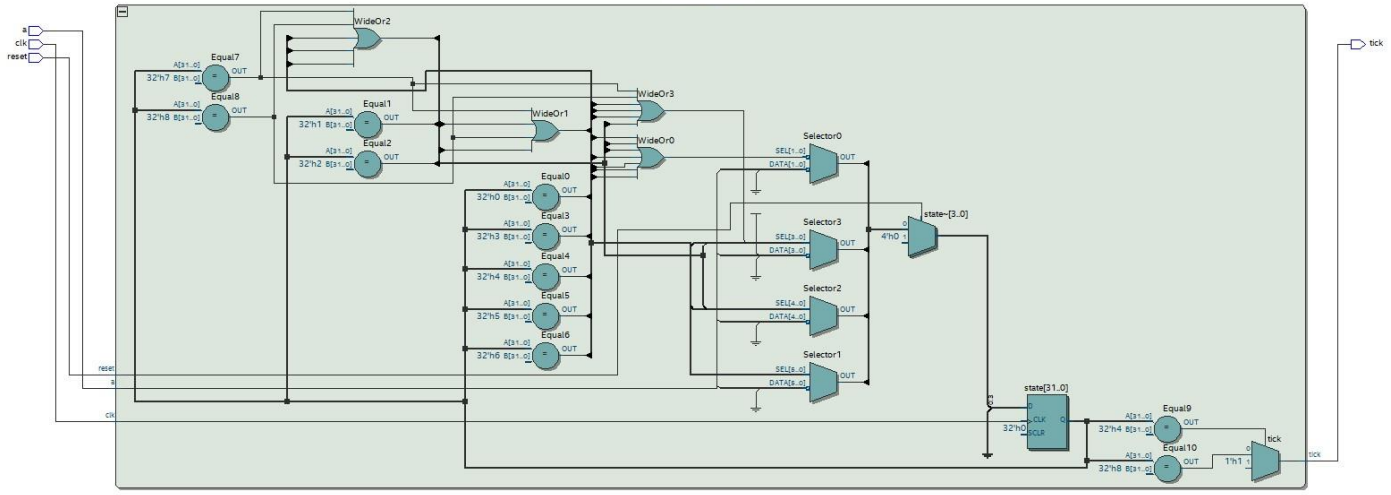
```
/*Hazırlayanlar: Senanur Ağaç – Beyza Duran
Lab6 problem 1 Testbench dosyası */
`timescale 1ns/1ps
module tb_lab6_g15_p1 ();
logic clk,reset,a;
logic tick;
lab6_g15_p1 uut (clk,reset,a,tick);
always
begin
clk=0;#5;clk=1;#5; end
always
begin
a=1; #60; a=0 ; #60; end
initial begin
reset=0;
#1000;
$stop ;
end
endmodule
```

Devrenin kodları kutucuklar içerisinde verilmiştir. Öncelikle modüle içinde portlar tanımlanmıştır. Daha sonra enum içerisinde stateler tanımlanmıştır {s0,s1,s2,s3,s4,s5,s6,s7,s8}. Daha sonra always_ff bloğu içinde her clock transition da state'in durumunu belirleyecek nextstate state'e atanmıştır. (Her clock rising edge de).

Devrenin Clock transition döngüsü s4 stateinde yinne a=1 gelirse s4 stateinde kalınır. Aynı durum s8 durumunda için de geçerlidir.s8 stateinde iken bir kez daha ~a gelirse (a==0) state s8 olarak kalınır.

Şekil 2: Dalga şeması

Yukarıda ki simülason çıktısında a önce 4 clock rising edge de bir durumundadır ve 4. Clock rising edge de tick sinyali bir olmuştur. 5. clock rising edge durumda da yine a 1'e eşit olduğu için tick yine 1 olarak kalmıştır.



RTL ŞEMASI

Slow 1200mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	572.74 MHz	250.0 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

Şekil 3:Fmax değeri

Analysis & Synthesis Resource Utilization by Entity

<<Filter>>

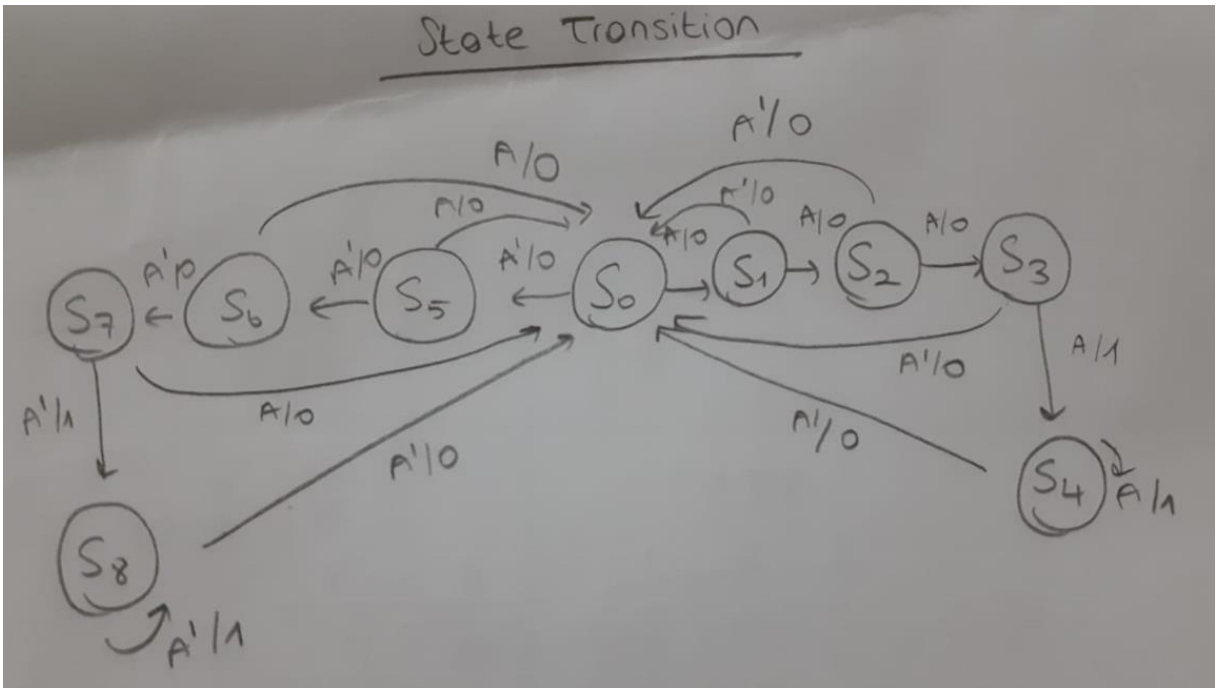
	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Memory Bits	UFM Blocks	DSP Elements	DSP 9x
1	|problem_top	9 (0)	4 (0)	0	0	0	0
1	|lab6_g15_p1 :inst|	9 (9)	4 (4)	0	0	0	0

Analysis & Synthesis Resource Utilization by Entity

<<Filter>>

M Blocks	DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC blocks	Full Hierarchy Name	Entity Name	Library Name
	0	0	0	4	0	0	|problem_top	lab6_g15_p1	work
	0	0	0	0	0	0	|problem_t..g15_p1 :inst	lab6_g15_p1	work

Şekil 4: Ultimizasyon Raporu



Şekil 5 : State Transition

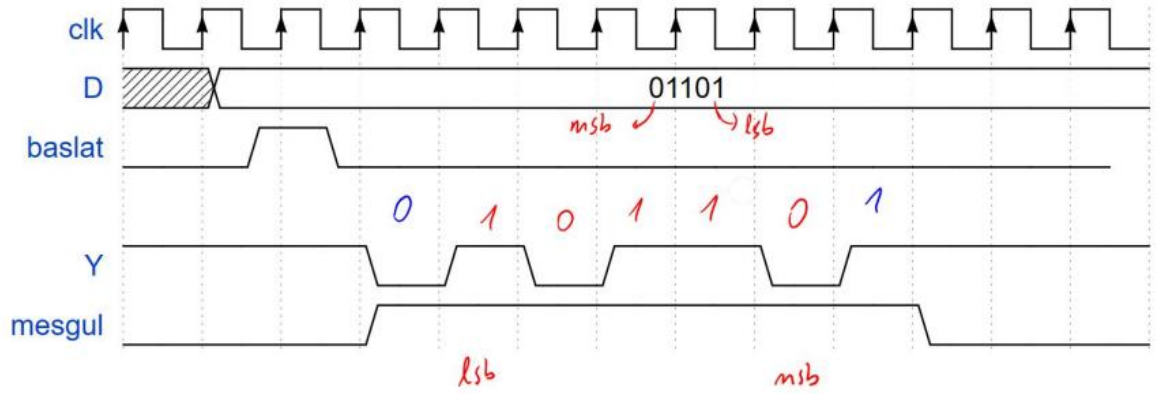
2.1.3 Sonuçların Yorumu

İstenilenlere göre FSM devresi oluşturularak birer adet input ve output içeren devre oluşturulmuştur ve buna uygun kodlar yazılmıştır. Kodun testi simüle edildildi. Simülasyon sonucu incelendiğinde sonucun doğru olduğu görülmüştür.

2.Problem - 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 6 da verilmiştir.



Şekil 6

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

```
module p2 (
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.

.2.1 Problemin Çözümü

```
/* Hazırlayanlar: Senanur Ağaç – Beyza Duran
Lab6 problem 1 Sıralayıcı
*/
module lab6_g15_p2 (
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,nextstate ;
always_ff @(posedge clk)
if(reset) state<=s0;
else state<=nextstate ;
always_comb
begin
case(state)
s0:
if(baslat==1)
nextstate=s1;
else
nextstate=s0;
s1:
if(!baslat)
nextstate=s2;
else
nextstate=s1;
s2:
```

1

```
if(!baslat)
nextstate=s3;
else
nextstate=s2;
s3:
if(!baslat)
nextstate=s4;
else
nextstate=s3;
s4:
if(!baslat)
nextstate=s5;
else
nextstate=s4;
s5:
if(!baslat)
nextstate=s6;
else
nextstate=s5;
s6:
if(!baslat)
nextstate=s7;
else
nextstate=s6;
nextstate=s6;
s7:
```

2

```
if(!baslat)
nextstate=s0;
else
nextstate=s7;
default:
nextstate=s0;
endcase
end
always_comb
begin
if(state==s0)
y=1;
else if(state==s1) begin
y=0;
mesgul=0;
end
else if(state==s2) begin
y=d[0];
mesgul=1;
end
else if (state==s3)begin
y=d[1];
mesgul=1;
```

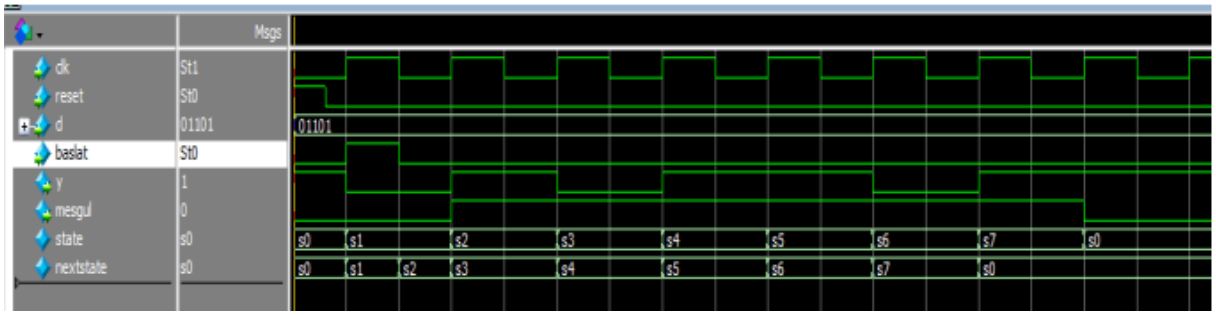
3

```
end
else if (state==s4)begin
y=d[2];
mesgul=1;
 end
else if (state==s5)begin
y=d[3];
mesgul=1;
end
else if (state==s6)begin
y=d[4];
mesgul=1;
 end
else if (state==s7) begin
y=1;
mesgul=1;
end
else begin
y=0;
mesgul=1;
end
end
endmodule
```

4

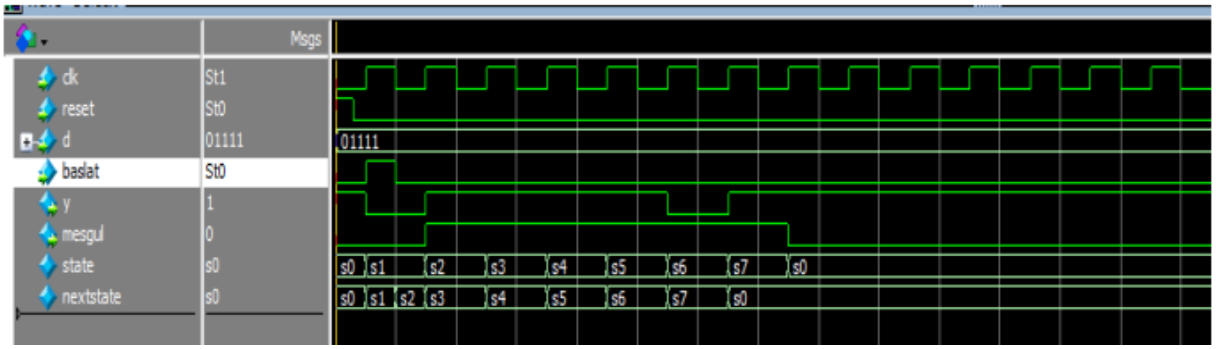
```
/*Hazırlayanlar: Senanur Ağaç – Beyza Duran
Lab6 problem 2 Testbench dosyası */
`timescale 1ns/1ps
module lab6_g15_p2_tb ();
logic clk,reset;
logic [4:0] d;
logic baslat;
logic y;
logic mesgul;
lab6_g15_p2 uut(clk,reset,d,baslat,y,mesgul);
always begin
clk=0; #5 ; clk=1 ; #5 ;
end
initial
d=5'b01101;
initial begin
baslat=0; #5;
baslat=1 ; #5;
baslat=0;
end
initial begin
reset=1; #3;
reset=0;
end
initial begin
#1500;
$stop;
end
endmodule
```

Sorunun isterlerinde belirtildiği gibi giriş olarak clk,en,reset ve D inputları atanmış olup çıkış olarak ise meşgul ve Y outputları atanmıştır. Koddaki always_ff bloğundan sonra her clock rising edge olduğu zamanlarda state=nextstate durumu ayarlanmıştır.Buna bağlı olarak örneğin state =s0 olduğu zaman başlat sinyali gelmişse nextstate=s1 olmuştur.Eğer başlat sinyali gelmemişse yani başlat==0 ise nextstate=s0 olarak kalmıştır. Bu durum s7 ye kadar devam edip . s7 durumunda ise başlat sinyali gelmemişse nextstate=s0 olmuştur. Eğer başlat sinyali gelmiş ise nextstate=s7 olarak atama yapılmıştır. Daha sonraki always_comb bloğunda ise y ve meşgul sinyalinin hangi durumlarda 1, 0 olacağı veya y=d[i] olacağı belirlenmiştir. Örneğin state=s2 durumunda y=d[0] olmuştur.

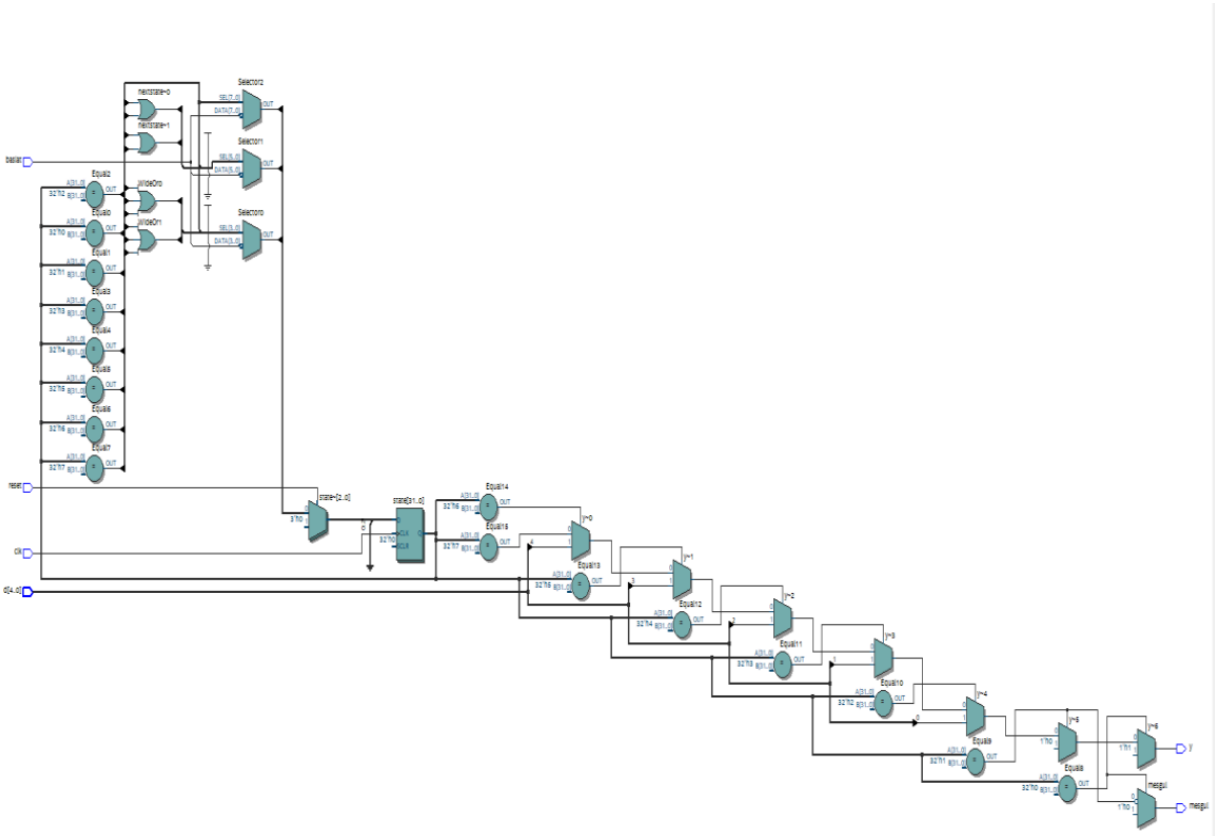


Şekil 7: Dalga şeması

Yukarıdaki simülasyon çıktısında da görüleceği gibi başlat sinyali bir clock cycle süre boyunca aynı olduktan sonra d inputu y outputuna aktarılmaktadır. Aktarma başlamadan önce y çıkışına bir clock cycle süresi kadar 0 ve aktarıldıktan osnra ise bie clock cycle boyunca 1 sinyali verilmektedir.



Şekil 8:Devrede d=01111 sinyali için deneme yapılmıştır. Ve sonuç başarılıdır.

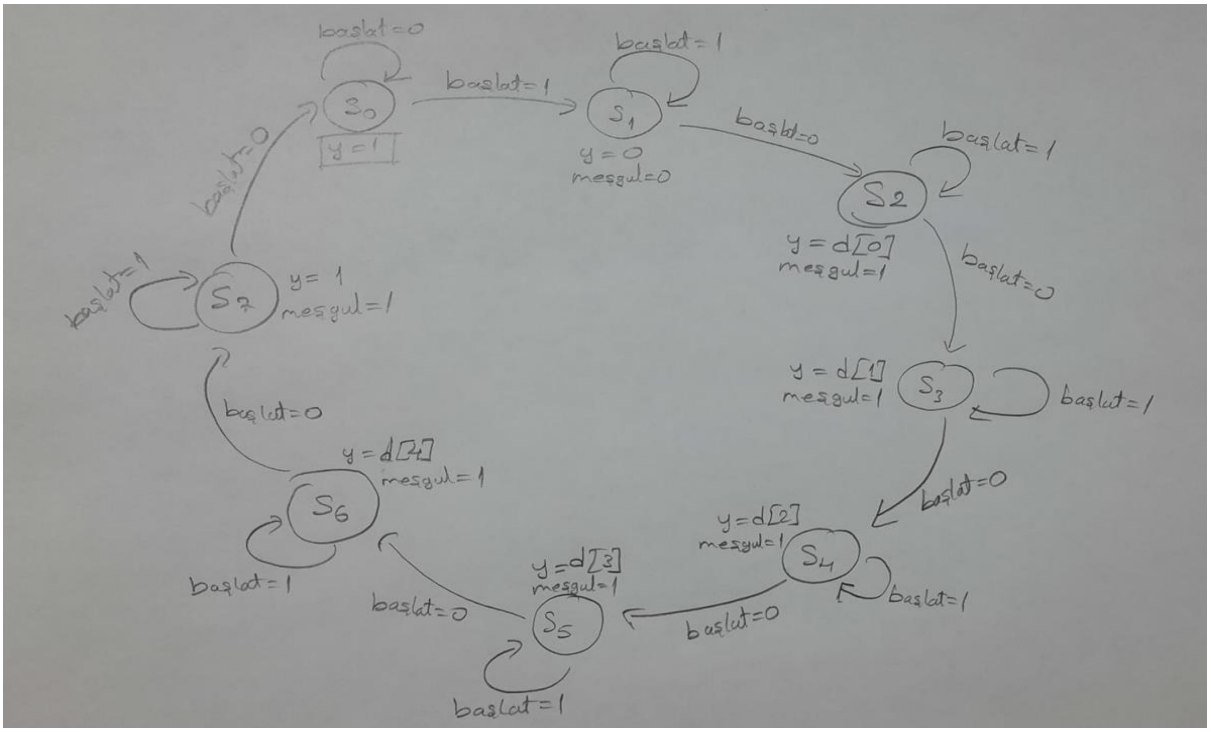


Şekil 9:RTL şeması

	Resource	Usage
1	Estimated Total logic elements	10
2		
3	Total combinational functions	10
4	Logic element usage by number of LUT inputs	
1	-- 4 input functions	7
2	-- 3 input functions	1
3	-- <=2 input functions	2
5		
6	Logic elements by mode	
1	-- normal mode	10
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[0]
15	Maximum fan-out	7
16	Total fan-out	53
17	Average fan-out	1.61

	Fmax	Restricted Fmax	Clock Name	Note
1	718.91 MHz	250.0 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

Şekil10:Maksimum frekans (FMAX)



Şekil 11: Problem 2 için State Transition

Sonuçların Yorumu

İstenilenlere göre FSM devresi oluşturularak dört adet input ve iki adet output içeren devre oluşturulmuştur ve buna uygun kodlar yazılmıştır. Kodun testi simüle edilmiştir. İkinci soruda bazı statelerin yazımında zorluklar çekilse de internet araştırmaları yardımıyla çözüm bulunmaya çalışılmıştır.