GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

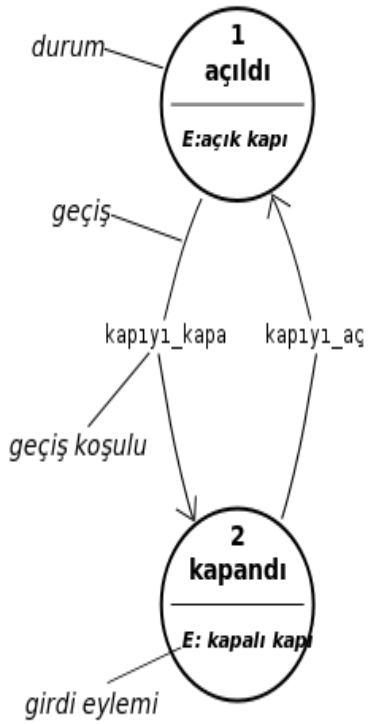
LAB 0x6 Deney Raporu

SONLU OTOMATLAR

Hazırlayanlar
1) 1901022038 – SELEN ERDOĞAN

1. Giriş

1- SONLU OTOMAT NEDİR?



Şekil 1 Sonlu Durum Makinası

Sonlu durum makinası (veya **sonlu durum otomatı veya basitçe durum makinası**), sınırlı sayıda durumdan, durumlar arası geçişlerden ve eylemlerin birleşmesiyle oluşan davranışların bir modelidir.Durum geçmiş hakkında bilgi saklar, örneğin başlangıçtan şu anki duruma kadar girdi değişimlerini gösterir. Geçiş durum değişimini gösterir ve geçişi sağlamak için yapılması gereken koşulla tanımlanır. Eylem belirli bir zamanda gerçekleştirilen etkinliğin tanımıdır. Birçok eylem tipi vardır:

***Giriş eylemi**
Bu eylem duruma geçerken gerçekleştirilir
***Çıkış eylemi**
Bu eylem durumdan çıkarken gerçekleştirilir
***Girdi eylemi**
Mevcut duruma ve girdi koşullarına bağlı gerçekleştirilen eylemdir
***Geçiş eylemi**
Belirli bir geçiş gerçekleştirilirken oluşan eylemdir

SDM durum çizgeleriyle (veya geçiş çizgeleriyle) temsil edilir (Bkz. Şekil 1). Bunun dışında çok sayıda durum geçiş tablo tipleri kullanılmaktadır. En çok karşılaşılan temsil aşağıda gösterilmiştir: mevcut durum (B)'de iken koşul (Y) gerçekleştiğinde sonraki durum (C) ortaya çıkar. Tüm eylemlerin bilgisi ancak dipnot kullanımıyla eklenebilmektedir. Tüm eylemlerin bilgisini içeren bir SDM tanımı durum tablolarını kullanarak mümkündür (Bkz. Sanal Sonlu Durum Makinası).

Durum Geçiş Tablosu

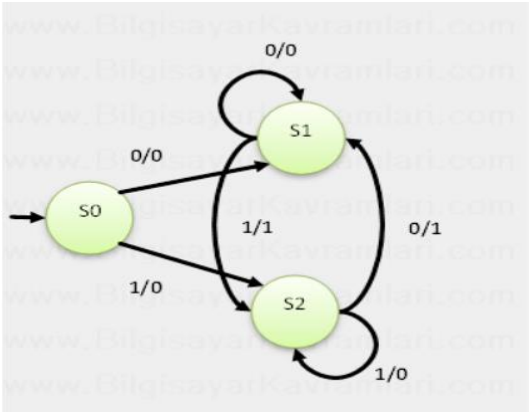
Mevcut Durum → Koşul	Durum A	Durum B	Durum C
Koşul X	...	...	...
Koşul Y	...	Durum C	...
Koşul Z	...	...	...

Tablo 1 Durum Geçiş Tablosu

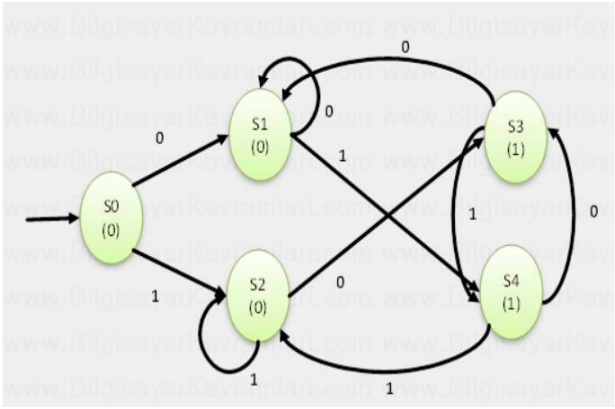
Burada gösterilen tepkisel sistemleri modellemeye ek olarak, sonlu durum makinaları çok farklı alanda önemlidir, bu alanlar elektrik mühendisliği, dilbilim, bilgisayar bilimleri, felsefe, biyoloji, matematik ve mantık olarak sayılabilir. Sonlu durum makinaları otomata teorisi ve hesaplama teorisinde çalışılan otomatların bir sınıfıdır. Bilgisayar bilimlerinde, sonlu durum makinaları uygulama davranışı, donanım sayısal sistemlerinin tasarımı, yazılım mühendisliği, ağ protokolleri ve hesaplama ve dillerin öğretilmesinde geniş ölçüde kullanılmaktadır.

2-MEALY ve MOORE MAKİNALARI

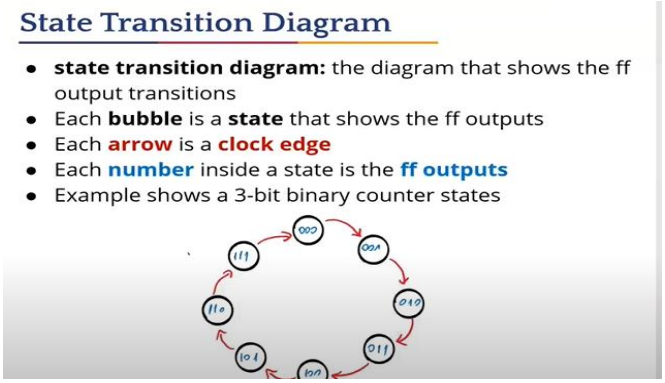
Bilgisayar bilimlerinde sıkça kullanılan sonlu durum makinelerinin (finite state machine, FSM veya Finite State Automaton , FSA) gösteriminde kullanılan iki farklı yöntemdir. Genelde literatürde bir FSM'in gösteriminde en çok moore makinesi kullanılır. Bu iki yöntem (mealy ve moore makinaları) sonuçta bir gösterim farkı olduğu için bütün mealy gösterimlerinin moore ve bütün moore gösterimlerinin mealy gösterimine çevrilmesi mümkündür.



Tablo 2 Mealy Makinesi



Tablo 3 Moore Makinesi



MEALY MOORE FARKI

Moore makinelerinde çıkış değerleri düğümlere (node) yazılırken, giriş değerleri kenarlar (edges) üzerinde
gösterilir. **Mealy** makinelerinde ise giriş ve çıkış değerleri kenarlar (edges) üzerinde aralarına bir taksim işareti (slash) konularak gösterilir. Örneğin 1/0 gösterimi, girişin 1 ve çıktının 0 olduğunu ifade eder.

1. PROBLEM

```
module lab6_1 ( input logic clk,reset,a,// girişler tanımlandı
output logic tick );// çıkışlar tanımlandı
typedef enum {s0,s1,s2,s3,s4,s5,s6,s7,s8} statetype ;
 statetype state,nextstate ;// enum oluşturuldu
always_ff @(posedge clk) // clock transition yapıldı
 if(reset) state<=s0;
else state<=nextstate ;
```

TABLO 1: Kod Başlangıcı

```
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
```

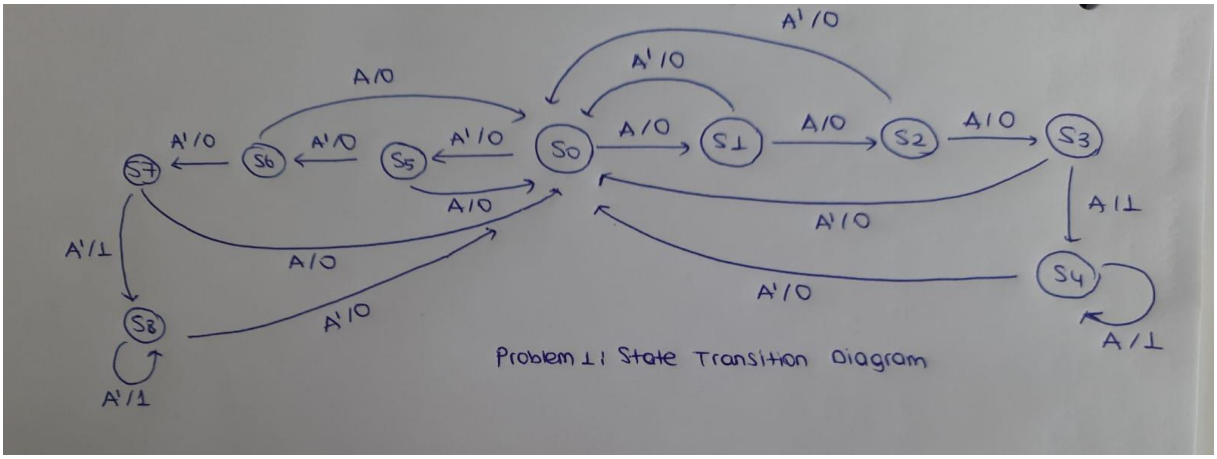
```
default:
nextstate=s0;
endcase
always_comb
if(state==s4)
tick=1;
else if(state==s8)
tick=1;
else
tick=0;
endmodule
```

Tablo 2 ve Tablo 3 ve Tablo 4: Kod Devamı

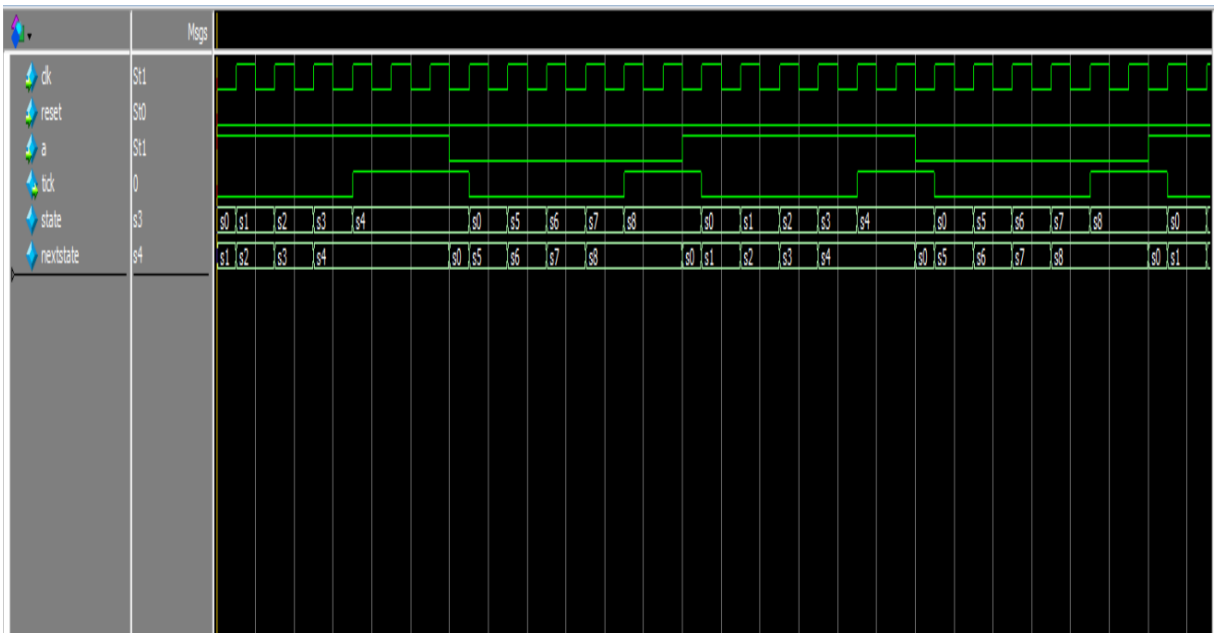
```
`timescale 1ns/1ps
module tb_lab6_1 ();
logic clk,reset,a;
logic tick;
lab6_1 uut (clk,reset,a,tick);
always begin
clk=0;#5;clk=1;#5;
end
always begin
a=1; #60;
a=0 ; #60;
end
initial begin
reset=0; #1000;
$stop ;
end
endmodule
```

Tablo 5: Testbench Dosyası

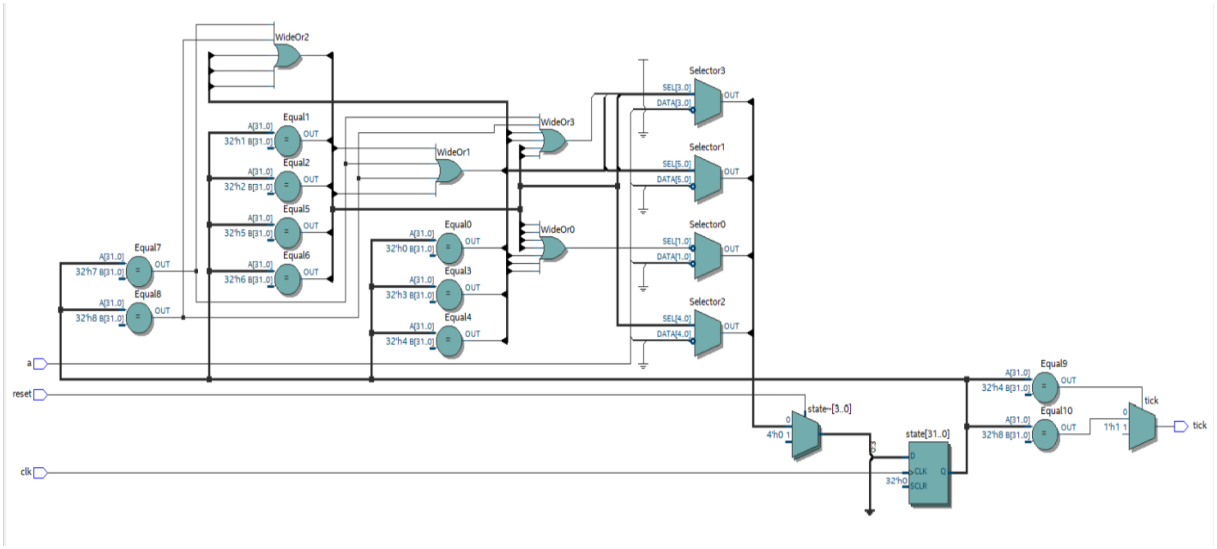
Devrenin giriş ve çıkış portları tanımlanmıştır. Enum fonksiyonu ile state durumları tanımlanmıştır. Nextstate ve state olarak iki adet durum belirlemesi yapılmıştır. Sonrasında always_ff bloğu içerisinde reset sinyali bire eşit ise state=s0 tanımlaması yapılmıştır. Eğer reset sinyali yok ise state = nextstate durum güncellemesi yapılmıştır. Always_comb bloğu içerisinde state=s0 ise, giriş sinyali a varsa nextstate=s1 olarak ayarlanmıştır. Eğer a'nın değili varsa nextstate=s5 olarak ayarlanmıştır. state =s1 durumunda ise a varsa nextstate =s2 yoksa nextstate=s0 olarak ayarlanmıştır. state =s2 durumunda ise a varsa nextstate =s3 yoksa nextstate=s0 olarak ayarlanmıştır. state =s3 durumunda ise a varsa nextstate =s4 yoksa nextstate=s0 olarak ayarlanmıştır. state =s4 durumunda ise a varsa nextstate =s4 yoksa nextstate=s0 olarak ayarlanmıştır.Şuana kadar state s0 dan state s5 e kadar belirtilen bütün durumlar 4 kere üst üste a sinyalinin bir gelmesi durumu için yazılmıştır.Dikkat edilmesi gereken bir nokta olarak state=s4 durumunda iken yeniden a sinyali 1 geldi ise tick sinyali gene 1 olmalıdır.Yani gene aynı state de (s4) kalmalıdır.Eğer sürekli 4 kere ~a gelirse ; State=s0 durumunda nextstate s5 durumuna geçer.s5 durumunda gene ~a gelirse ise nextstate= s6, gelmez ise nextstate=s0 state durumuna geri dönülür. S6 durumunda gene ~a gelirse ise nextstate= s7 ,gelmez ise nextstate=s0 state durumuna geri dönülür. S7 durumunda gene ~a gelirse ise nextstate= s8, gelmez ise s0 state durumuna geri dönülür. S8 durumunda ise gene ~a gelir ise state aynı nextstate=s8 olarak kalır .Eğer a =1 gelir ise nextstate=s0 olarak belirlenir.



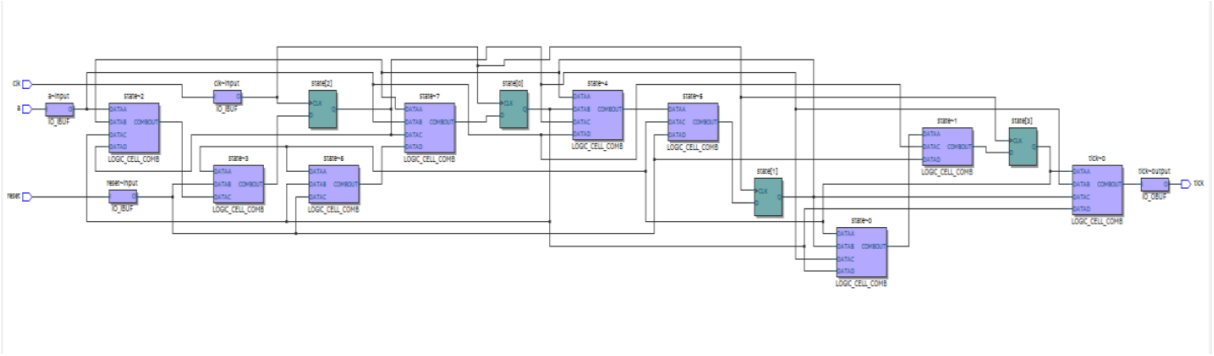
Şekil 4 Problem1: State Transtition Diagram



Şekil 5 Problem1: Dalga Şeması



Şekil 6 Problem1: RTL şeması



Şekil 7 Problem1: Post-Mapping

	Fmax	Restricted Fmax	Clock Name	Note
1	621.89 MHz	250.0 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

Şekil 8 Problem1: Fmax Değeri

Flow Status	Successful - Mon May 11 17:18:29 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	lab6_2
Top-level Entity Name	lab6_2
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	10 / 49,760 (< 1 %)
Total registers	4
Total pins	4 / 360 (1 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

Analysis & Synthesis Status	Successful - Mon May 11 17:18:07 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	lab6_2
Top-level Entity Name	lab6_2
Family	MAX 10
Total logic elements	9
Total registers	4
Total pins	4
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

	Resource	Usage
1	Estimated Total logic elements	9
2		
3	Total combinational functions	9
4	⌄ Logic element usage by number of LUT inputs	
1	-- 4 input functions	5
2	-- 3 input functions	4
3	-- <=2 input functions	0
5		
6	⌄ Logic elements by mode	
1	-- normal mode	9
2	-- arithmetic mode	0
7		
8	⌄ Total registers	4
1	-- Dedicated logic registers	4
2	-- I/O registers	0
9		
10	I/O pins	4
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	state[3]
15	Maximum fan-out	5
16	Total fan-out	45
17	Average fan-out	2.14

Şekil 9 Problem1:Utilization Raporu

2-PROBLEM 2

Shift Register Nedir?

Bu sıralı cihaz, girişlerinde bulunan verileri yükler ve daha sonra her saat döngüsünde bir kez çıkışına taşır veya "kaydırır", dolayısıyla **Shift Register** olarak adlandırılmaktadırlar.

Bir Shift Register temel olarak, her veri biti için bir tane olmak üzere, bir "0" veya "1" mantığı olan ve birbirine bağlanan birkaç tek bitlik "D-Tipi Veri Mandalından" oluşmaktadır. Bir veri mandalı, bir sonraki mandalın girişi olur ve bu böyle devam eder.
Veri bitleri, bir Shift Register içine veya dışına seri olarak, yani sol veya sağ yönden birbiri ardına olacak şekilde sıralanacaktır. Bunu yanı sıra paralel bir konfigürasyonda aynı anda birlikte de beslenebilmektedirler.

```
module lab6_2

( 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 ;
```

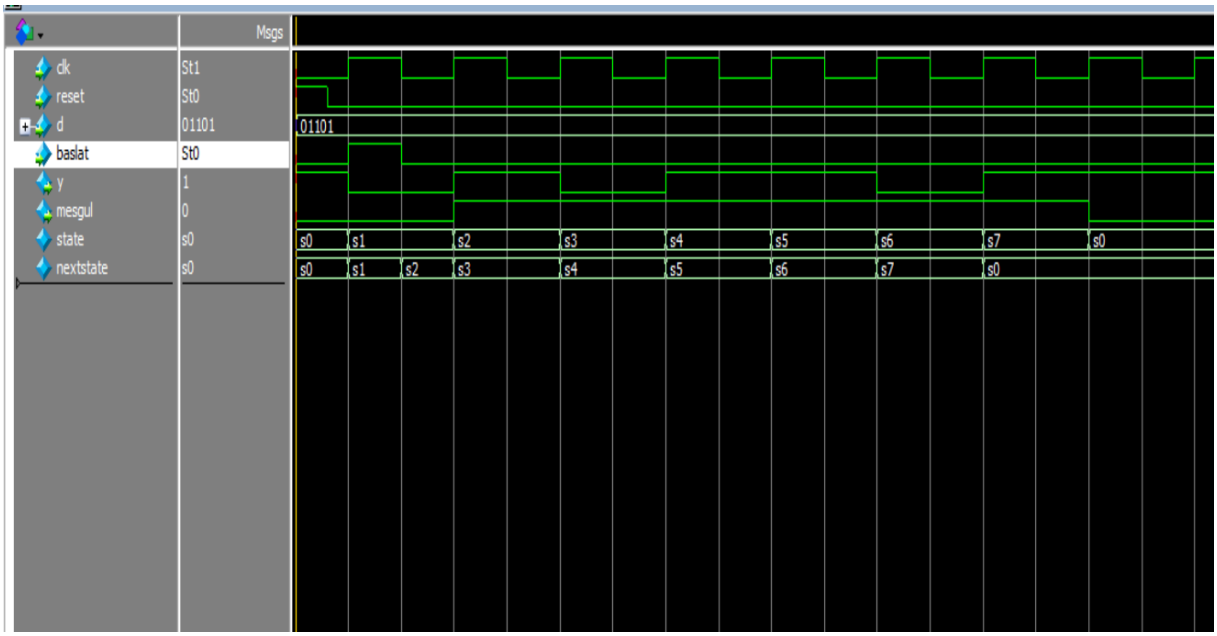
```
always_comb

begin case(state)

s0: if(baslat==1)

nextstate=s1;

else nextstate=s0;

s1: if(!baslat)

nextstate=s2;

else nextstate=s1;

s2: 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: 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;

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
```

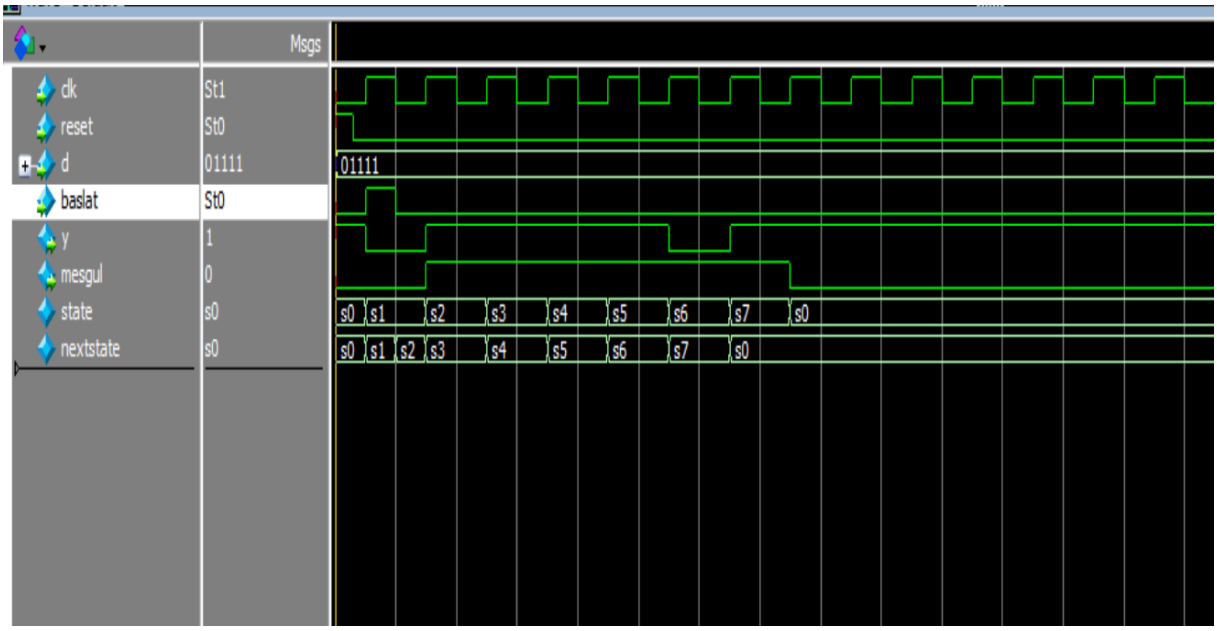
```
`timescale 1ns/1ps

module tb_lab6_2 ();

logic clk,reset;

logic [4:0] d;

logic baslat;

logic y;

logic mesgul;

lab6_2 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
```

Bu problemde giriş bölümünde söylenildiği gibi clk,reset,en ve D portları giriş olarak atanmıştır.(D portu 5 bitliktir).Output olarak ise meşgul ve Y portları deney föyünde istenildiği gibi belirlenmiştir. Devredeki ilk always_ff bloğunda state transition ayarlanmıştır. Yani her clock rising edge olduğu zaman state=nextstate olmuştur.Devrede daha sonra gelen always_comb bloğunda ise deney föyünde belirtilen talimatlar doğrultusunda if-else ve case deyimleri kullanılmıştır. State =s0 olduğu zaman başlat sinyali gelmişse nextstate=s1 olmuştur.Eğer başlat sinyali gelmemişse yani başlat==0 ise nextstate=s0 olarak kalmıştır. S1 durumunda ise başlat sinyali gelmemişse nextstate=s2 olmuştur. Eğer başlat sinyali gelmiş ise nextstate=s1 olarak atama yapılmıştır. S2 durumunda ise başlat sinyali gelmemişse nextstate=s3 olmuştur. Eğer başlat sinyali gelmiş ise nextstate=s2 olarak atama yapılmıştır. S3 durumunda ise başlat sinyali gelmemişse nextstate=s4 olmuştur. Eğer başlat sinyali gelmiş ise nextstate=s3 olarak atama yapılmıştır. S4 durumunda ise başlat sinyali gelmemişse nextstate=s5 olmuştur. Eğer başlat sinyali gelmiş ise nextstate=s4 olarak atama yapılmıştır. S5 durumunda ise başlat sinyali gelmemişse nextstate=s6 olmuştur. Eğer başlat sinyali gelmiş ise nextstate=s5 olarak atama yapılmıştır. S6 durumunda ise başlat sinyali gelmemişse nextstate=s7 olmuştur. Eğer başlat sinyali gelmiş ise nextstate=s6 olarak atama yapılmıştır. 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.

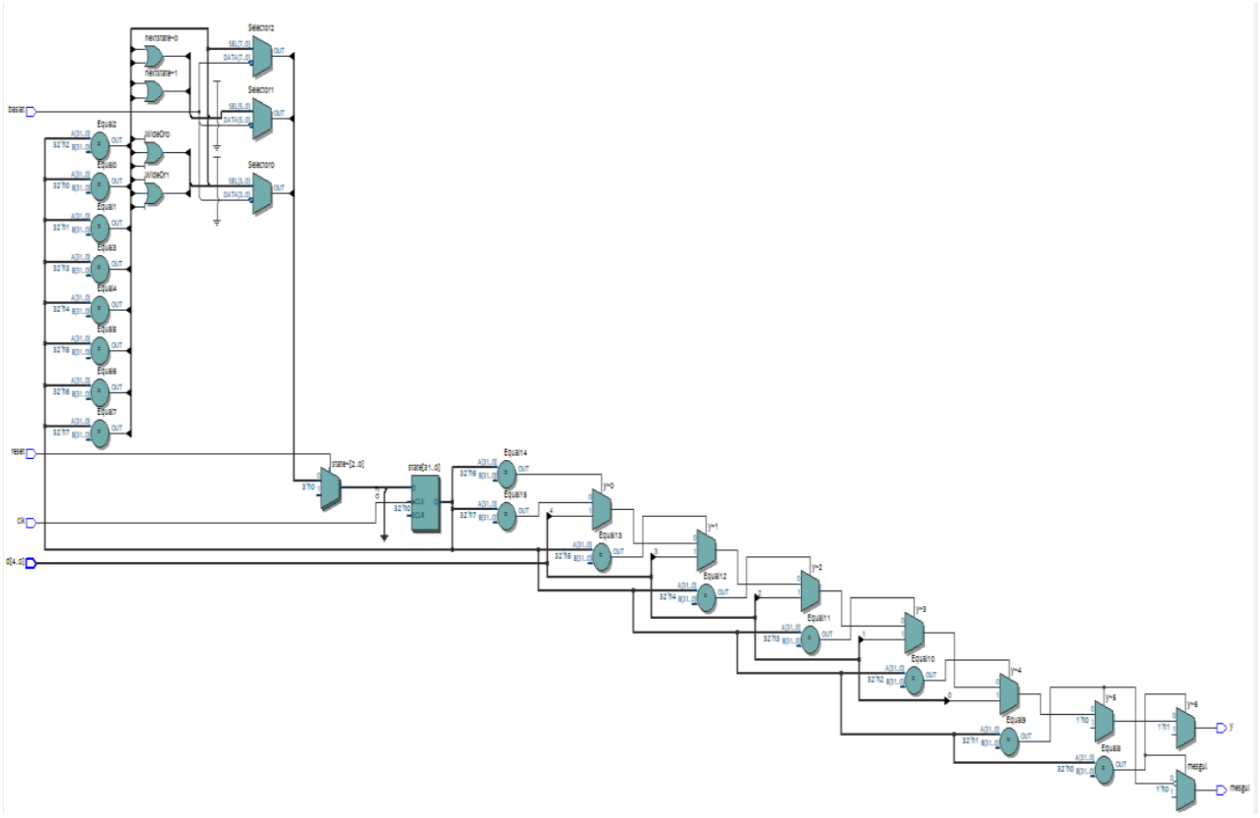


Şekil 10 Problem2: Dalga Şeması

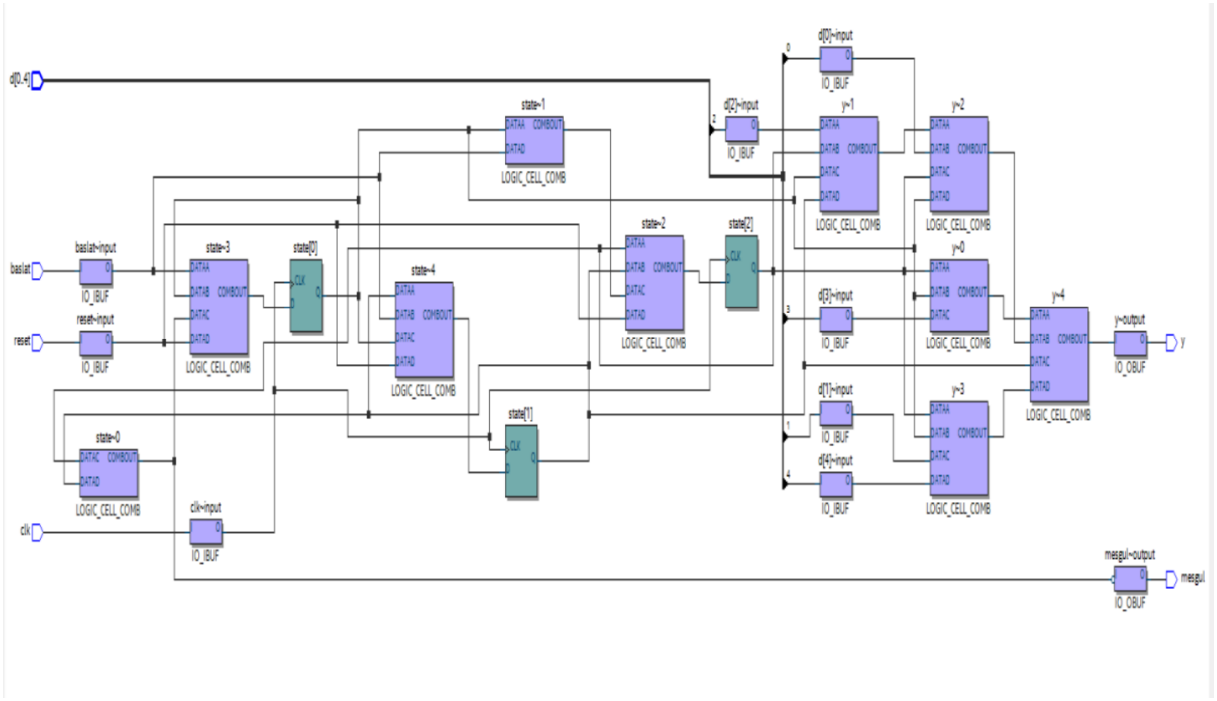
Devre d=01111 sinyali için tekrar denenmiştir.



Şekil 11 Problem2: Dalga Şeması



Şekil 12 Problem2: RTL Şeması



Şekil 13 Problem2: Post-Mapping

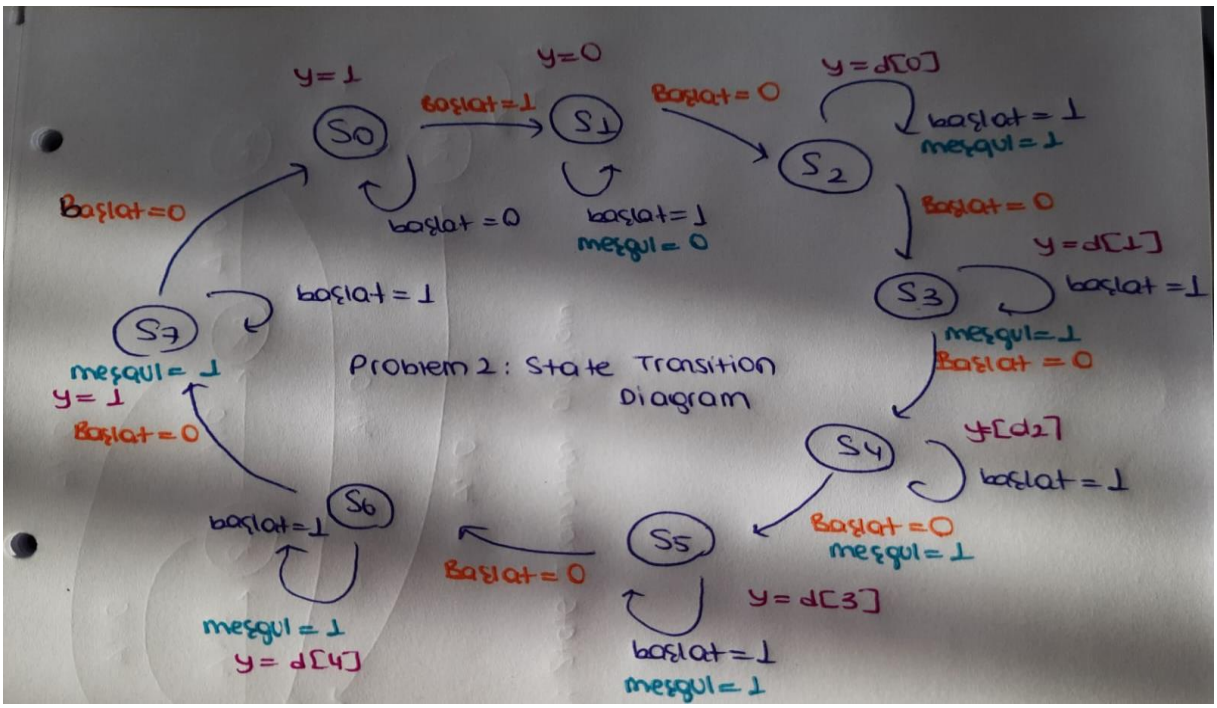
Analysis & Synthesis Status	Successful - Mon May 11 18:21:27 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	lab6_2
Top-level Entity Name	lab6_2
Family	MAX 10
Total logic elements	10
Total registers	3
Total pins	10
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

	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

Şekil 14 Problem2: Utilization Raporu

	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)

Şekil 15 Problem2: Fmax



Şekil 16 Problem2: State Transition Diagram

Özet:

Mealy Moore durum diyagramları arasındaki farklar öğrenildi.

- Moore modelinde çıktılar yalnızca durumun bir fonksiyonudur. Mealy modelinde çıktılar durumun ve girişlerin bir fonksiyonudur.
- Mealy devreler Moore devrelerine göre daha küçük alan gerektirir.
- Mealy Moore devrelerine dönüştürülebilir.
- Mealy devrelerin çıkışları giriş değerleri değişitiği anda güncelleşir.
- Moore durum modeline uygun devrelerde çıkışlar saat sinyali ile senkronizedir.

Referanslar:

[Ardışıl Devreler-11] Sözlü Tariften Moore Tipi Ardışıl Devre Tasarımı (Örnekli Anlatım)
https://www.youtube.com/watch?v=tQRWKwNNCa0&list=PLp2H2NvRDf2Tm9P3uGt-g2TwSR0WiSRS3&index=13

[Ardışıl Devreler-12] Sözlü Tariften Mealy Tipi Ardışıl Devre Tasarımı (Örnekli Anlatım)
https://www.youtube.com/watch?v=6POeiguQA0s&list=PLp2H2NvRDf2Tm9P3uGt-g2TwSR0WiSRS3&index=11

Sadi Evren Şeker:https://bilgisayarkavramlari.com/2011/01/26/mealy-ve-moore-makineleri-mealy-and-moore-machines/

3. Ders - Mealy ve Moore Durum Diyagramları
https://www.youtube.com/watch?v=B_AbWjeBOLY

Sonlu Durum Makinası: Sonlu durum makinesi - Vikipedi (wikipedia.org)