



GEBZE TEKNİK ÜNİVERSİTESİ
ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 0x7 Deney Raporu

SONLU OTOMATLAR - II

Hazırlayanlar
1) 1901022025- AYŞE SERRA ŞİMŞEK
2) 1901022038- SELEN ERDOĞAN

1. GİRİŞ:

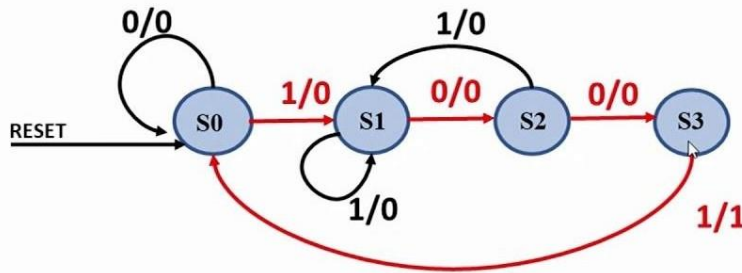
Bu deneyde Sequence Detector ve Counters devre tasarımı istenmiştir. Deneyde FSM kullanılmıştır. İki devre içinde Verilog donanım dili kullanılarak modelsim üzerinde kod yazılıp, Quartus Prime programında sentezlenerek RTL ve Post Mapping şemaları elde edilmiştir.

Deneye başlamadan önce sahip olunması gereken teorik bilgiler araştırılmıştır. Bu bağlamda geçen haftaki deneyin teorigine ek olarak;

Bir **Sequence Detector** (dizi detektörü), bir giriş bit dizisini alan ve hedef dizi algılandığında bir çıktı 1 üreten bir sıralı durum makinesidir. Bir Mealy makinesinde çıktı, mevcut duruma ve harici girdiye (x) bağlıdır. Bu nedenle, diyagramda çıktı, girdilerle birlikte durumların dışında yazılmıştır. Örnek olarak,

State diagram for sequence detector using Mealy Model

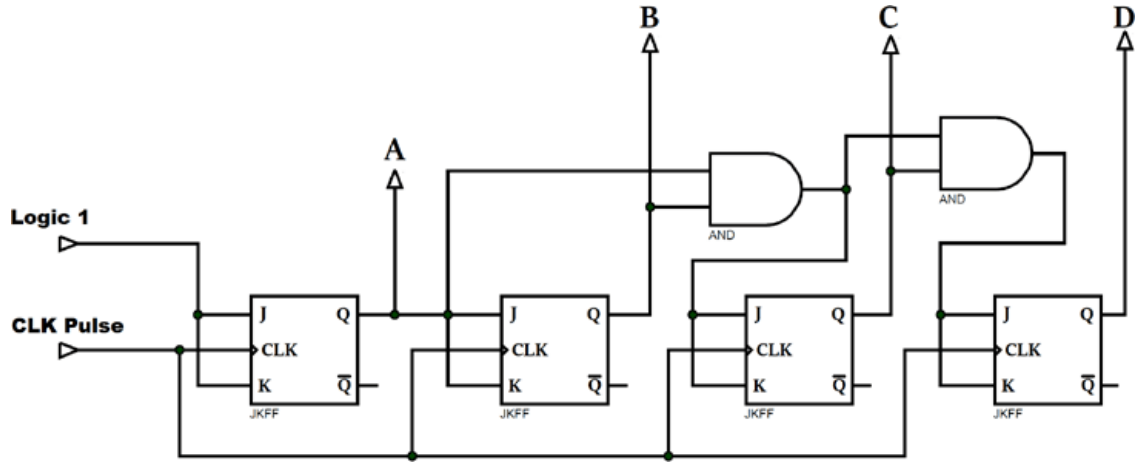
- State diagram to detect a sequence **1001** using Mealy Model (Non-overlapping)



Sıralı bir devre tasarımı için aşağıdaki gibi bir prosedür izlenebilir:

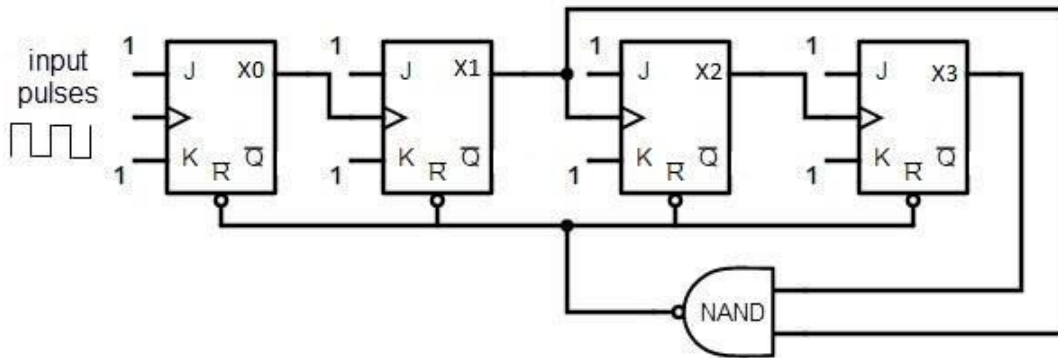
1. Devrenin durum diyagramı ve durum tablosu üretilir.
2. Durum diyagramındaki durumların sayısı sayılır (N olarak adlandırılır). $[2^{(P-1)} < N \leq 2^P]$ denklemi çözülerek gereken flip-flop sayısı hesaplanır. (P olarak adlandırılır.)
3. Her duruma benzersiz bir P-bit ikili sayı (durum vektörü) atanır. Genellikle, ilk durum = 0, sonraki durum = 1, vb.
4. State transition tablosu ve çıktı tablosu üretilir.
5. State transition tablosu, her flip-flop için bir tane olmak üzere P tablolarına ayrılır.
6. Kullanılacak flip-flop türlerine karar verilir. Emin olunmadığı takdirde, tüm JK'lar kullanılır.
7. Her bir flip-flop için girdi tablosu üretilir.
8. Tüm flip-flop'ların giriş durumunun ve mevcut durumunun fonksiyonlarına dayalı olarak her flip-flop için giriş denklemleri türetilir.
9. Denklemler tek bir yere yazılarak özetlenir.
10. Devre şeması çizilir.

Binary Senkron sayıcıda, tüm "flip-floplar"daki saat girişi aynı kaynağı kullanır ve aynı anda aynı saat sinyalini yaratır. Aynı kaynaktan aynı anda aynı saat sinyalini kullanan sayaca Senkron sayıcı denir.



Senkron Sayaç Tasarımı

Bir **BCD sayacı**, uygulanan bir saat sinyali ile 0'dan önceden belirlenmiş bir sayıya kadar sayan 4 bitlik ikili sayaçlardan biridir. Sayım önceden belirlenen sayım değerine ulaştığında tüm flip-flop'ları sıfırlar ve tekrar 0'dan saymaya başlar. Bu tip sayaç 4 JK flip flop kullanılarak tasarlanır ve 0'dan 9'a kadar sayar ve sonuç dijital olarak temsil edilir. biçim. 9 (1001) sayısına ulaştıktan sonra sıfırlanır ve yeniden başlar.



BCD Sayaç Tasarımı

2. PROBLEMLER:

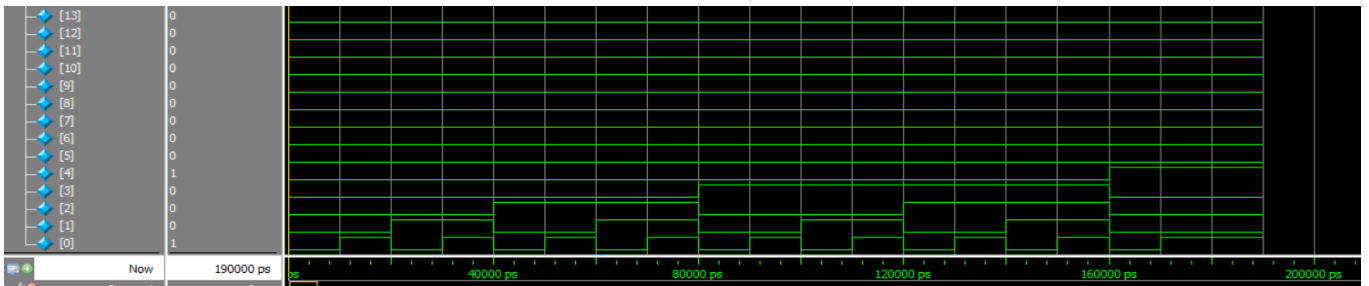
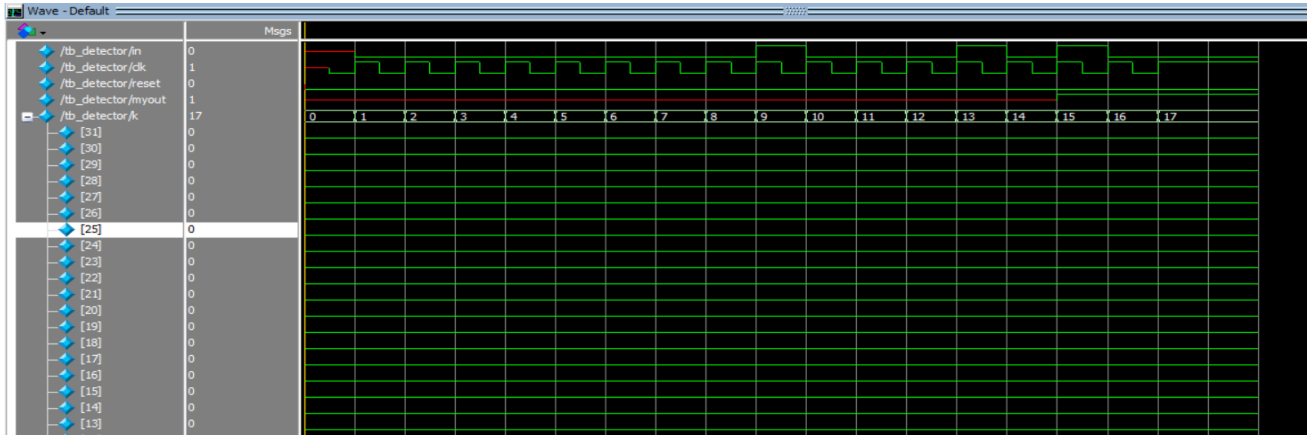
2.1. Problem Problem 1 - Sequence Detector

<pre> module detector (input logic in,clk,reset, output logic myout); logic out; typedef enum {s0,s1,s2} statetype ; statetype state,nextstate ; always_ff @(posedge clk) if(reset) state<=s0; else state<=nextstate ; always_comb begin case(state) s0: begin if(in)begin nextstate=s1; end else begin nextstate=s0; end end s1: begin if(in)begin nextstate=s0; end else begin nextstate=s2; end end </pre>	<pre> s2: begin if(in)begin nextstate=s0; out=1; end else begin nextstate=s0; end end default: nextstate=s0; endcase end always_comb begin myout=out; end endmodule </pre>
---	--

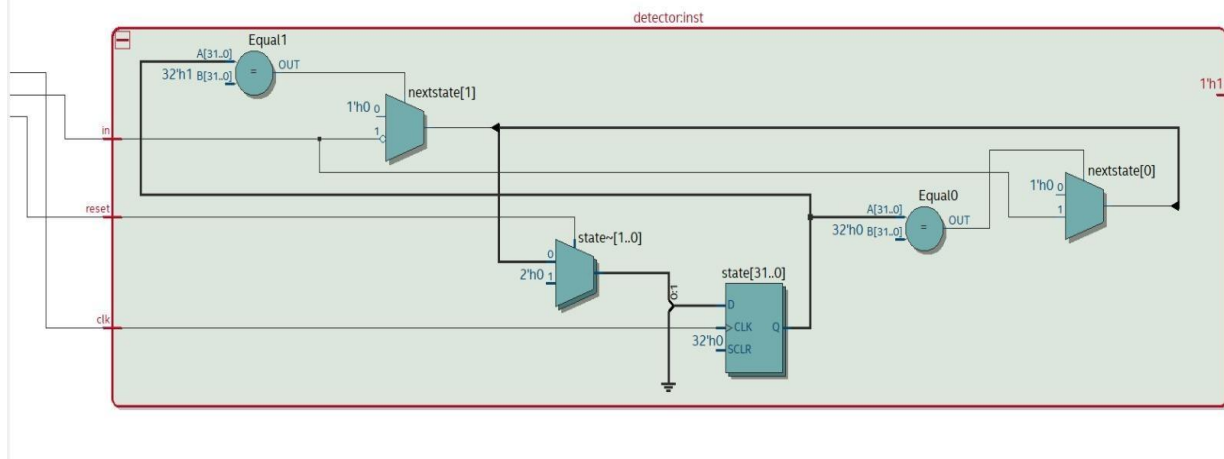
..sv dosyası

```
`timescale 1ns / 1ps  
  
module tb_detector();  
logic in;  
logic clk;  
logic reset;  
logic myout;  
  
detector sel(in,clk,reset,myout);  
  
integer k=0;  
  
initial  
begin  
  
reset=0;  
  
    for(k=0; k < 17; k=k+1)  
    begin  
        #5; clk=0;  
        #5; clk=1;  
        in=$urandom_range(0,1);  
    end  
  
    #20;  
  
    $stop;  
  
end  
endmodule
```

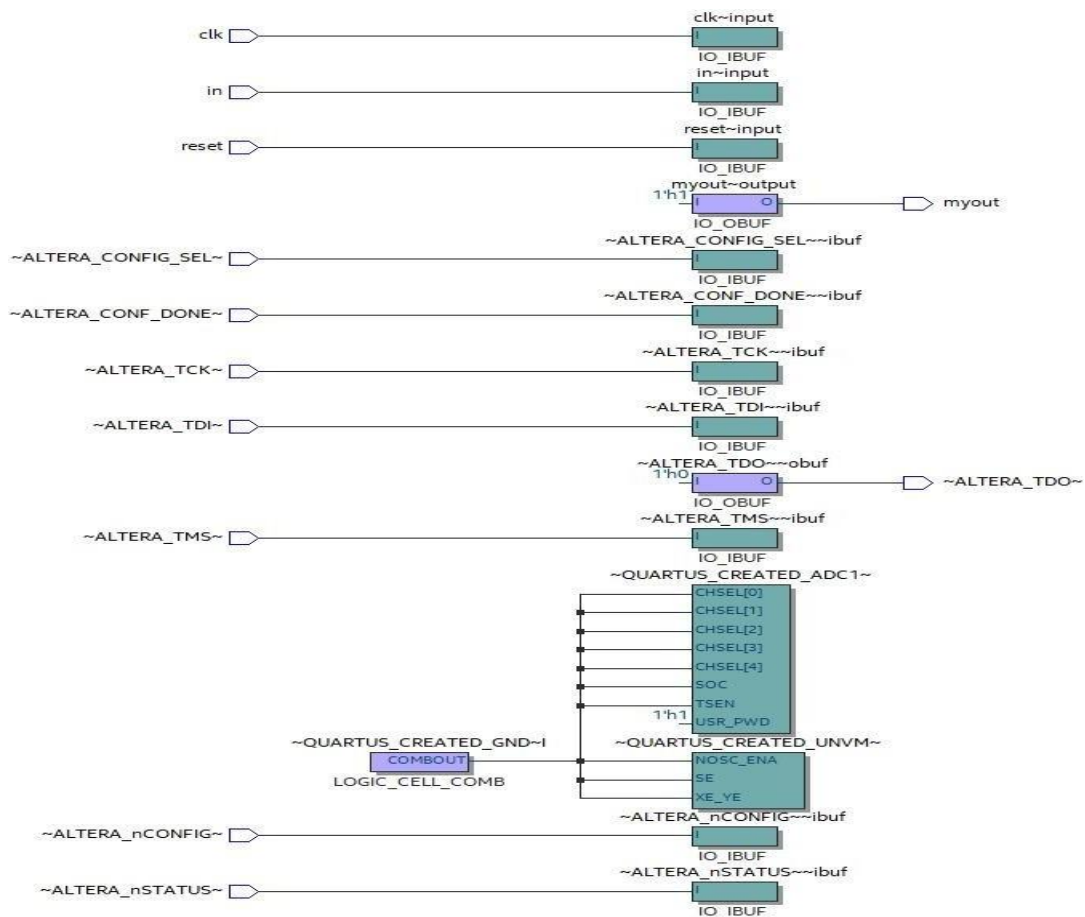
tb.sv dosyası



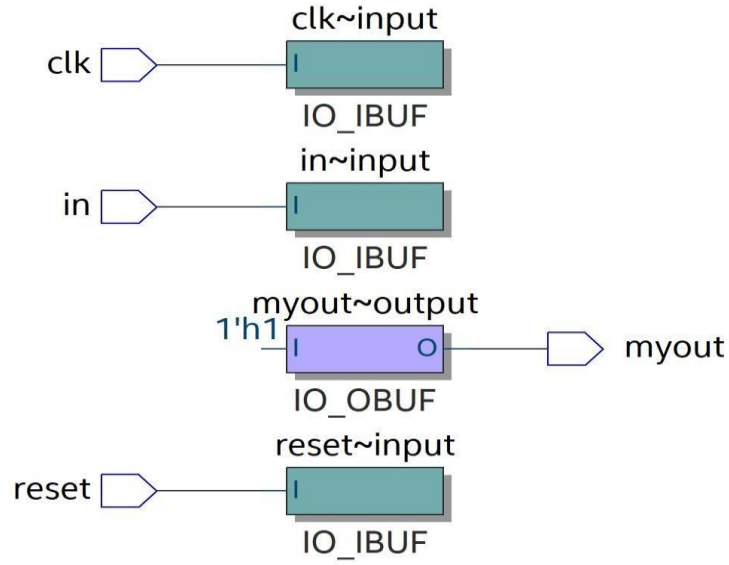
Şekil 1 Problem 1 Dalga Şeması



Şekil 2 Problem 1 RTL Şeması



Şekil 3 Problem 1 Post-Mapping



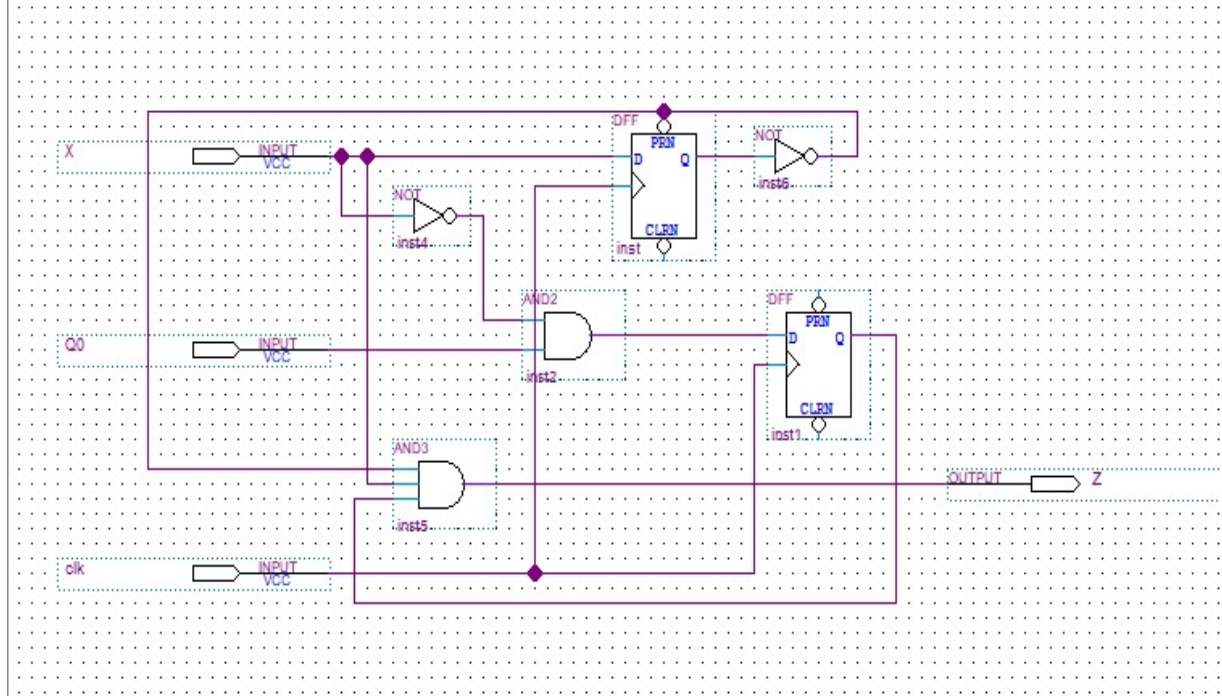
Şekil 4 Problem 1 Post-Fitting

Flow Summary	
<<Filter>>	
Flow Status	Successful - Sat May 14 10:23:06 2022
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	detector_top
Top-level Entity Name	detector_top
Family	MAX 10
Device	10M08DAF484C8G
Timing Models	Final
Total logic elements	1 / 8,064 (< 1 %)
Total registers	0
Total pins	4 / 250 (2 %)
Total virtual pins	0
Total memory bits	0 / 387,072 (0 %)
Embedded Multiplier 9-bit elements	0 / 48 (0 %)
Total PLLs	0 / 2 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 1 (0 %)

Şekil 5 Problem 1 Utilization Raporu

<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	665.78 MHz	250.0 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

Şekil 6 Problem 1 Fmax



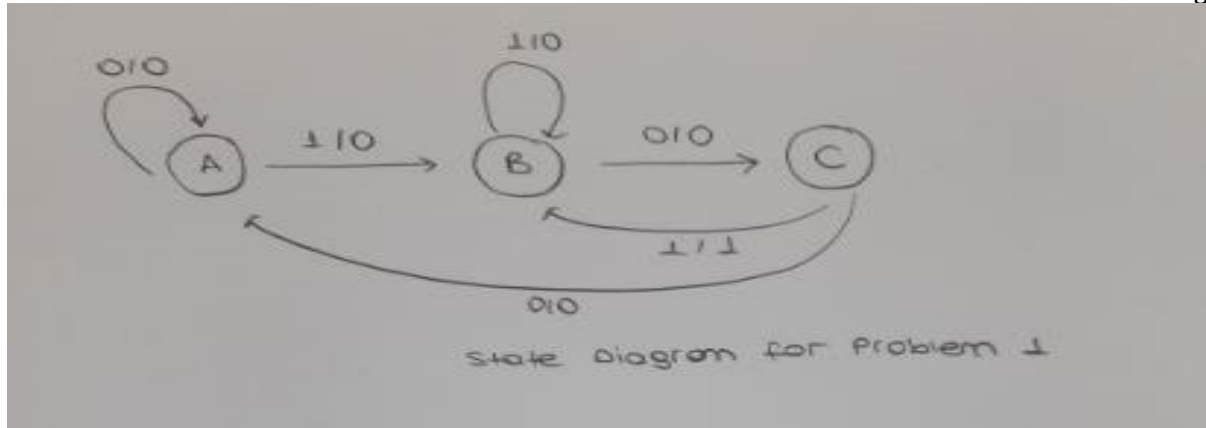
Şekil 7 Problem 1 Devre Şematiği

PS	I/P	NS	O/P	Flip Flop	
				D ₁	D ₀
00	0	00	0	0	0
00	1	01	0	0	1
01	0	10	0	1	0
01	1	01	0	0	1
10	0	00	0	0	0
10	1	10	1	0	1

Q ₁ \ Q _{0x}	00	01	11	10
0	0	1	1	0
1	0	1	X	X

D₀ = X

Şekil 1 State Table



Şekil 8 State Diagram for Problem 1

State Table			
PS	I/P	NS	O/P
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	1	B	1
C	0	A	0

For Problem 1

Şekil 9 State Table for Problem 1

2.2. Problem Problem 2 – Counters

<pre> module detector (input logic clk,reset, output logic [3:0] myout); logic [3:0] count; </pre>	<pre> s8: begin count= 4'b1000; nextstate=s9; end s9: begin </pre>
--	---

```

typedef enum
{s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15}
statetype ;

statetype state,nextstate ;

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

always_comb
begin

case(state)

s0: begin
count= 4'b0000;
nextstate=s1;
end

s1: begin
count= 4'b0001;
nextstate=s2;
end

s2: begin
count= 4'b0010;
nextstate=s3;
end

```

```

count= 4'b1001;
nextstate=s10;
end

s10: begin
count= 4'b1010;
nextstate=s11;
end

s11: begin
count= 4'b1011;
nextstate=s12;
end

s12: begin
count= 4'b1100;
nextstate=s13;
end

s13: begin
count= 4'b1101;
nextstate=s14;
end

s14: begin
count= 4'b1110;
nextstate=s15;

```

```
s3: begin
count= 4'b0011;
nextstate=s4;
end
```

```
s4: begin
count= 4'b0100;
nextstate=s5;
end
```

```
s5: begin
count= 4'b0101;
nextstate=s6;
end
```

```
s6: begin
count= 4'b0110;
nextstate=s7;
end
```

```
s7: begin
count= 4'b0111;
nextstate=s8;
end
```

```
end
s9: begin
count= 4'b1001;
nextstate=s10;
end
```

```
s15: begin
count= 4'b1111;
nextstate=s0;
end
```

```
default:
nextstate=s0;
endcase
```

```
end
always_comb
begin
myout=count;
end
endmodule
```

.sv dosyası

```

\timescale 1ns / 1ps

module tb_detector();
logic in;
logic clk;
logic reset;
logic myout;

detector sel(in,clk,reset,myout);

integer k=0;

initial
begin
    reset=0;

    for(k=0; k < 17; k=k+1)
    begin
        #5; clk=0;
        #5; clk=1;
        in=$urandom_range(0,1);
    end

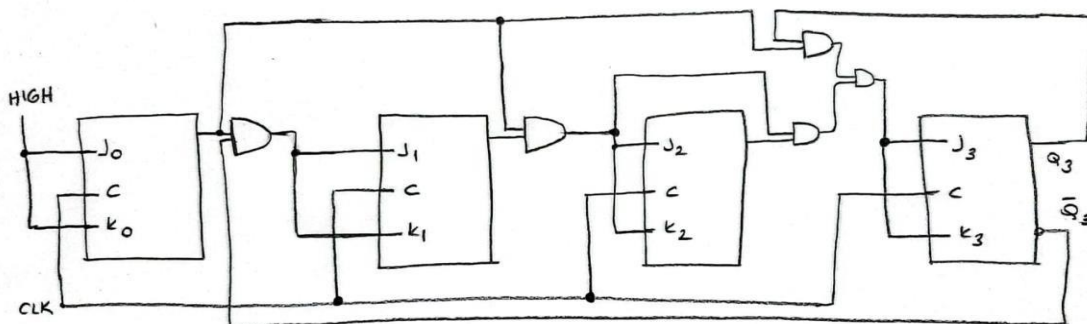
    #20;

    $stop;

end
endmodule

```

tb.sv dosyası



Şekil 2 BCD counter devre şeması

Present State (Q3 Q2 Q1 Q0)	Next State (Q3+ Q2+ Q1+ Q0+)	D3	D2	D1	D0
0000	0001	0	0	0	1
0001	0010	0	0	1	0
0010	0011	0	0	1	1
0011	0100	0	1	0	0
0100	0101	0	1	0	1
0101	0110	0	1	1	0
0110	0111	0	1	1	1
0111	1000	1	0	0	0
1000	1001	1	0	0	1
1001	1010	1	0	1	0
1010	1011	1	0	1	1
1011	1100	1	1	0	0
1100	1101	1	1	0	1
1101	1110	1	1	1	0
1110	1111	1	1	1	1
1111	0000	0	0	0	0

D3

Q1 Q0 \ Q3 Q2	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	1	0	1
10	1	1	1	1

www.visifacts.com

$$D3 = Q3Q2' + Q3Q0' + Q3Q1' + Q3'Q2Q1Q0$$

D2

Q1 Q0	00	01	11	10
Q3 Q2	00	0	1	0
01	1	1	0	1
11	1	1	0	1
10	0	0	1	0

$$D2 = Q2Q0' + Q2Q1' + Q2'Q1Q0$$

D1

Q1 Q0	00	01	11	10
Q3 Q2	00	0	1	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

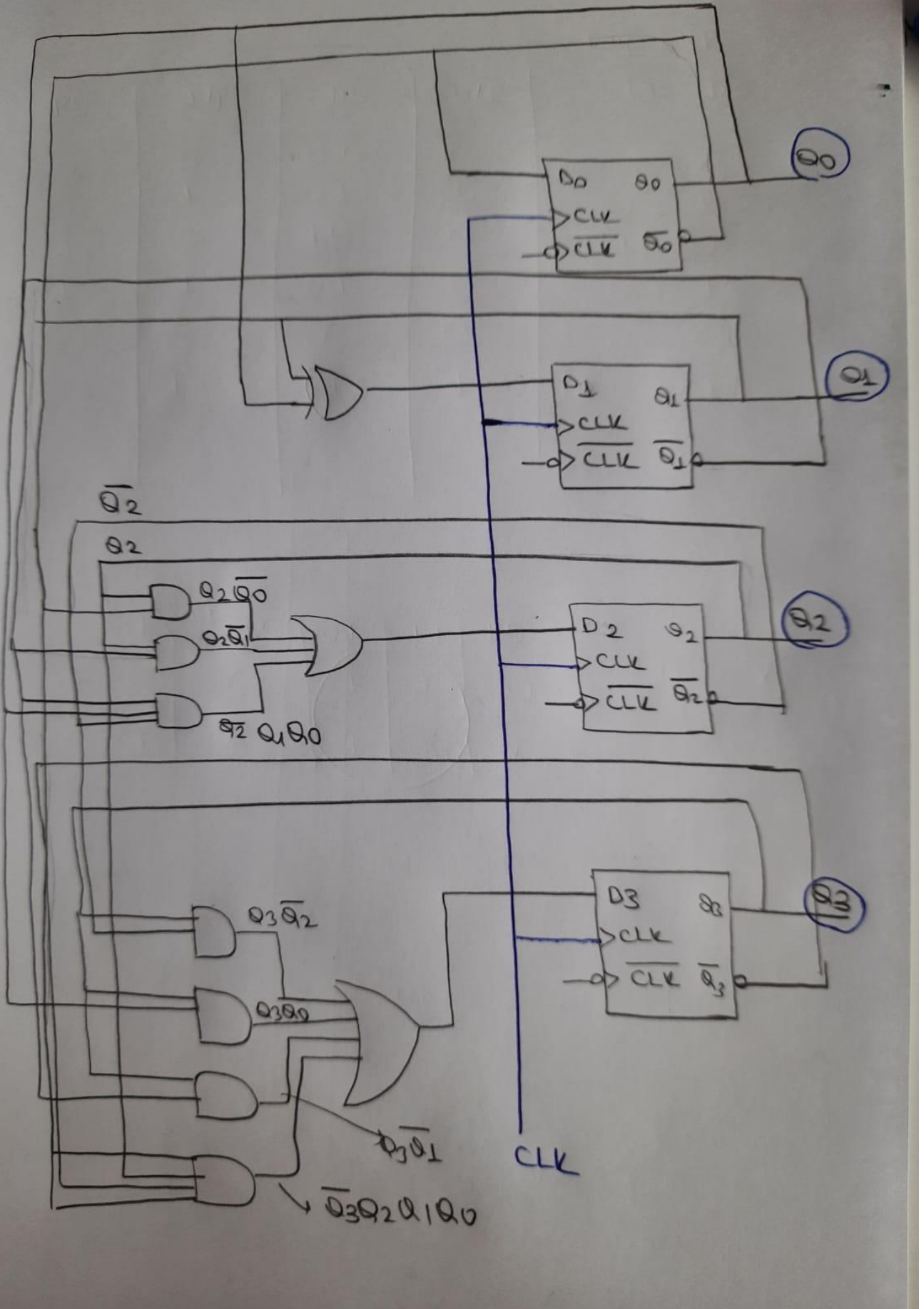
$$D1 = Q1'Q0 + Q1Q0' = Q1 \oplus Q0$$

D0

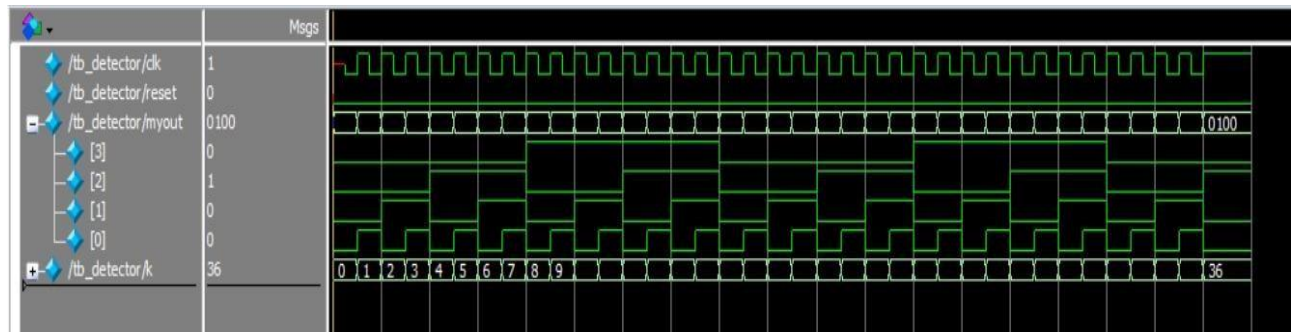
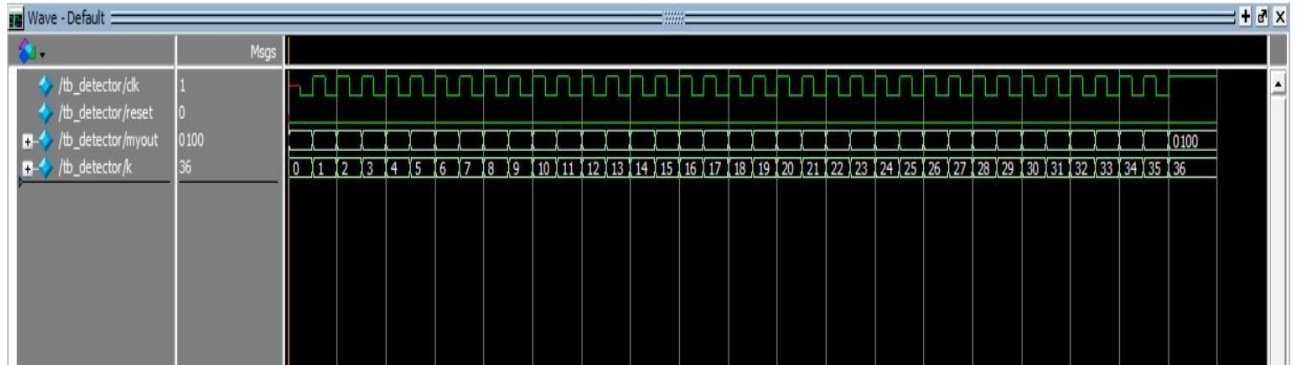
Q1 Q0	00	01	11	10
Q3 Q2	00	1	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

$$D0 = Q0'$$

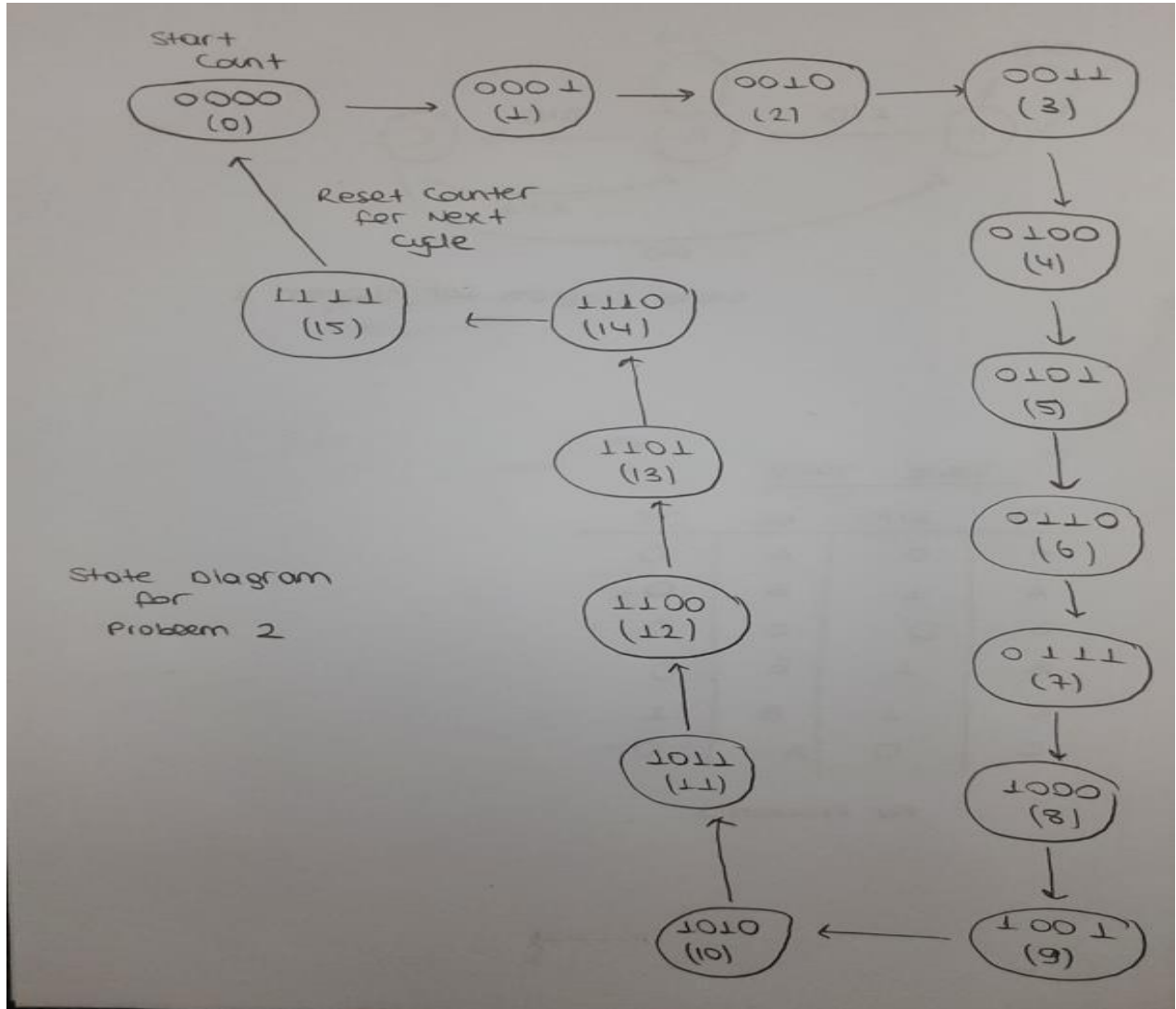
Şekil 3 Problem 2



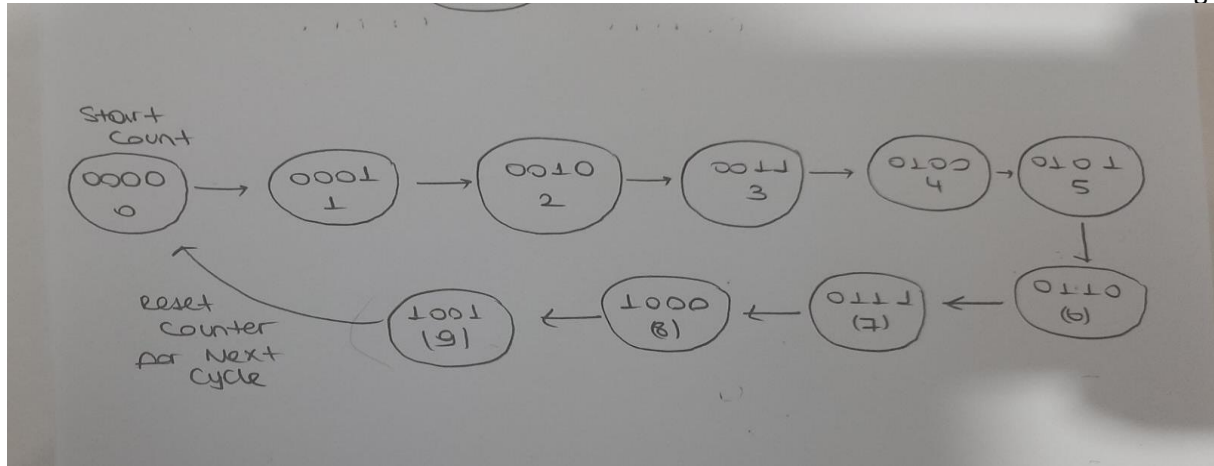
Şekil 4 D flip-flop kullanarak 4bit counter devre şeması



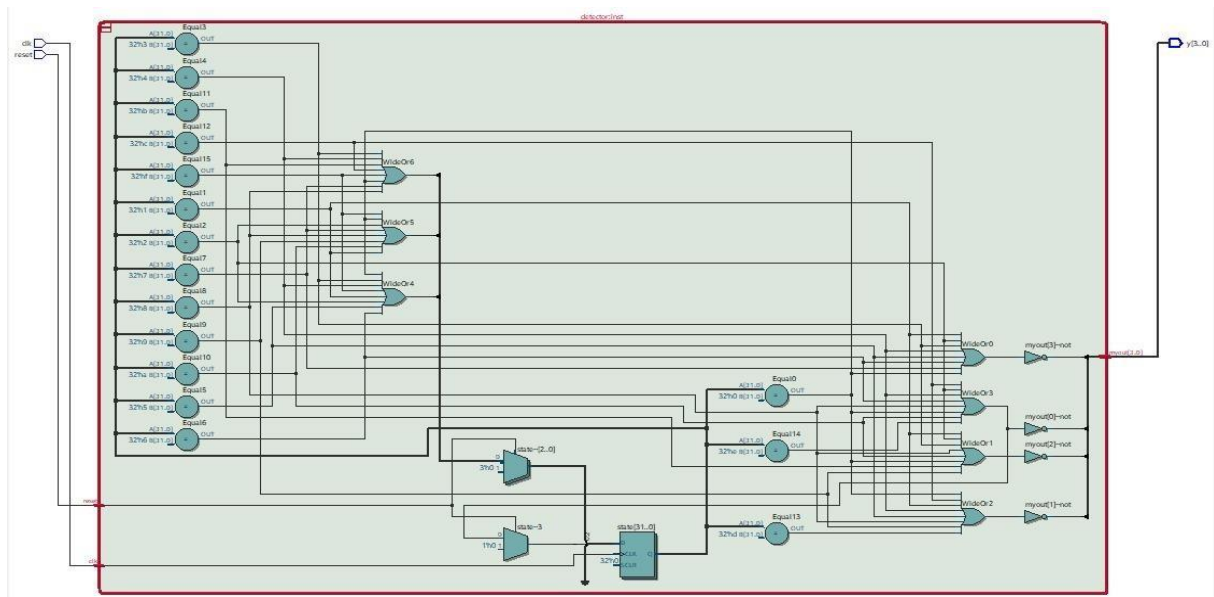
Şekil 10 Problem 2 Dalga Şeması



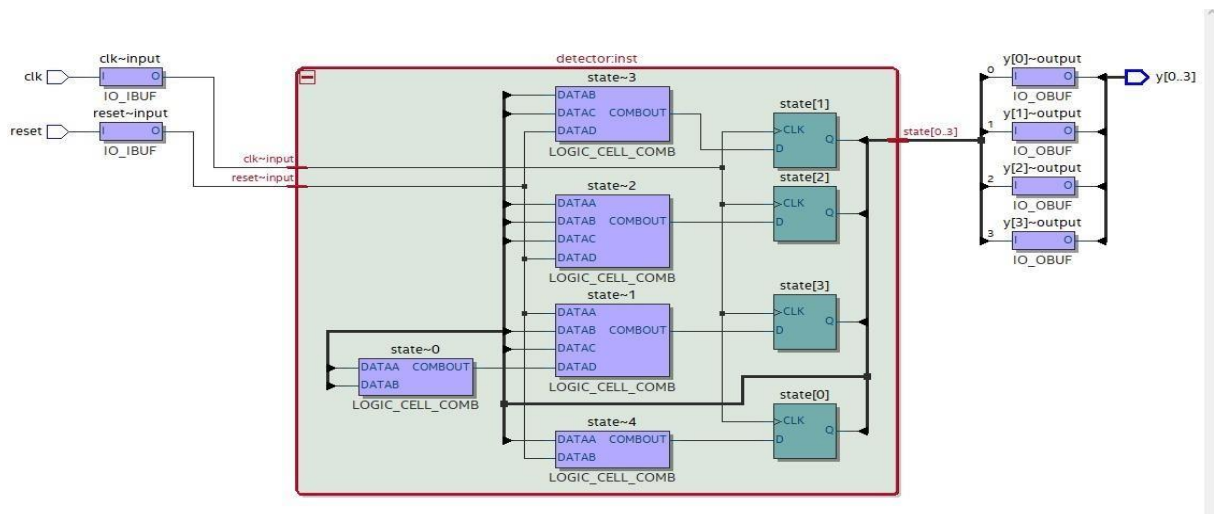
Şekil 11 Problem 2 State Diagram 4 bitlik Binary sayıcı için



Şekil 12 Problem 2 BCD senkron sayıcı için state diagram



Şekil 13 Problem 2 RTL Şeması



Şekil 14 Problem 2 Post-Mapping

Analysis & Synthesis Resource Utilization by Entity				
<<Filter>>				
	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Memory Bits
1	▼ P7_top	5 (0)	4 (0)	0
1	detector:inst	5 (5)	4 (4)	0

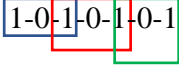
Analysis & Synthesis Resource Utilization by Entity							
<<Filter>>							
UFM Blocks	DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC blocks	Full
0	0	0	0	6	0	0	P7_top
0	0	0	0	0	0	0	P7_top

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimated Total logic elements	5
2		
3	Total combinational functions	5
4	▼ Logic element usage by number of LUT inputs	
1	-- 4 input functions	2
2	-- 3 input functions	1
3	-- <=2 input functions	2
5		
6	▼ Logic elements by mode	
1	-- normal mode	5
2	-- arithmetic mode	0
7		
8	▼ Total registers	4
1	-- Dedicated logic registers	4
2	-- I/O registers	0
9		
10	I/O pins	6
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	detec...te[0]
15	Maximum fan-out	5
16	Total fan-out	33
17	Average fan-out	1.57

Slow 1200mV OC Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	715.82 MHz	250.0 MHz	clk	limit due to minimum period restriction (r

Şekil 15 Problem 2 Utilization Raporu ve Fmax Değeri

3.Sonuç

Deneyde bulunan 1. Problemde, Sequence Detector (Sıralı dizi dedektörü) tasarımı istenmiştir. Bu tasarım gerçekleştirilirken föyde belirtildiği gibi her 1-0-1 dizisi için 1 çıkışı, diğer durumlarda 0 çıkışı elde edilmesi gerektiği söylenmiştir. Ayrıca belirtilen önemli bir nokta, oluşan dizinin iç içe geçtiği durumların da dahil edilmesi gerektiğidir. Örn. 

Bu gereksinimler dikkate alınarak state diyagram oluşturulmuştur. 2 flip flop ile devre kurulmuş ve test edilmiştir. Böylece iç içe geçen 101 dizilerini tespit edilebildiği görülmektedir.

2.Problemde, 0-9 arası sayım yapan BCD counter, problem 1 deki gibi diyagram ve state table oluşturularak flip flop girişleri bulunmuş ve flip floplar kullanılarak devre elde edilmiştir. Aynı şekilde 4 bit counter flip floplar kullanılarak elde edilmiştir. Testbench ile test edildiğinde istenilen çıktılar alınmıştır. BCD sayıcı, bir saat sinyali uygulamasında ona kadar sayabilen özel bir dijital sayaç ürünüdür. Bu durum state diagram yapısında da gösterilmiştir.

4.Referanslar

<https://studytronics.weebly.com/sequence-detectors.html>

<https://www.stickyminds.com/article/state-transition-diagrams>

<https://www.cs.umd.edu/~meesh/cmsc311/clin-cmsc311/Lectures/lecture29/counter.pdf>

<https://circuitdigest.com/tutorial/synchronous-counter>

<https://www.elprocus.com/bcd-counter-circuit-working/#:~:text=BCD%20or%20decade%20counter%20circuit%20is%20designed%20by%20using%20JK,decade%20counter%20is%20shown%20below.&text=From%20the%20figure%2C%20we%20observe,are%20connected%20to%20logic%201.>

https://www.electronics-tutorials.ws/counter/count_3.html

<https://www.youtube.com/watch?v=SXE8mSRmgIU>