GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 0x3 Deney Raporu

Donanım Tasarım Dillerine Giriş

Hazırlayanlar
1) 1901022038 -Selen Erdoğan
2) 1901022025 – Ayşe Serra Şimşek

1- GİRİŞ

- Bu deney kapsamında daha önceki deneylerde kullanılmayan ModelSim programı kullanılmıştır. Bu program kurulan lojik devrelerin simülasyonu için kullanılmaktadır. Devreleri kodlar yardımıyla kuracağız. Bu kodların program içinde anlam kazanması için verilog seçilmeli, yazılan kodlar uygun bir klasör ve dosyalara kaydedilmelidir. Bunun haricinde RTL şeması ve diğer bazı şemalar için Quartus Prime programından faydalanıldı. Bu iki program için bakıldığı zaman ModelSim programı karmaşık devreler için daha kullanışlı olmaktadır. Bu şekilde karmaşık devreler kodlar yardımıyla daha basit hale gelmektedir. Ayrıca kod üzerinde yapılan değişiklikler ile (#2 yazmak gibi) gecikmeler gözlemlenmiştir. Kurduğumuz devrede varsa bir hatanın simülasyon ekranında nasıl göründüğü gözlemlenmiştir. Son olarak oluşturulan bazı devreler quartus prime ile sentezlenerek RTL ve Post-Fitting şemaları nasıl olduğu görülmüştür.
- Bu deneyin yapılması için DTD' nin ne olduğu bilinmelidir. DTD devrelerin tasarımını, işlemlerini ve simülasyonlarını içermektedir. Bu sayede herhangi bir donanım için örnekleme yapılabilmektedir. Fiziksel olarak ortaya bir donanım birimi koymadan önce tasarımcının sistemi test etmesini sağlamaktadır. DTD ile yazılmış bir kod sentezleyici yazılım tarafından işlenmektedir. DTD hataların minimuma indirilmesi ve fiziksel olarak üretimin daha kesin bir şekilde yapılmasına yardımcı olmaktadır.

2- PROBLEMLER

$$Y = \overline{A}\ B$$

Denklem 1

2.1 Basit Bir Devre Yapısı Ve Simülasyonu

Verilen Boolean denklemini Verilog ile tasarlayıp Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemlememiz istenmiştir.

2.1.1 Kodlar

```
Ln#
1   module lab3_g0_p1 (
2    input  logic a, b,
3     output logic y
4   );
5
6       assign y = ~a & b;
7   endmodule
```

```
Ln#
1   `timescale 1ns/1ps`timescale 1ns/1ps
2   module tb_lab3_g0_p1 ();
3   logic a, b;
4   logic y;
5   lab3_g0_p1 uut0(.a(a), .b(b), .y(y));
6   initial begin
7   a = 0; b = 0; #10;
8   b = 1;        #10;
9   a = 1;        #10;
10  b = 0;        #10;
11                #20;
12  $stop;
13  end
14  endmodule
```

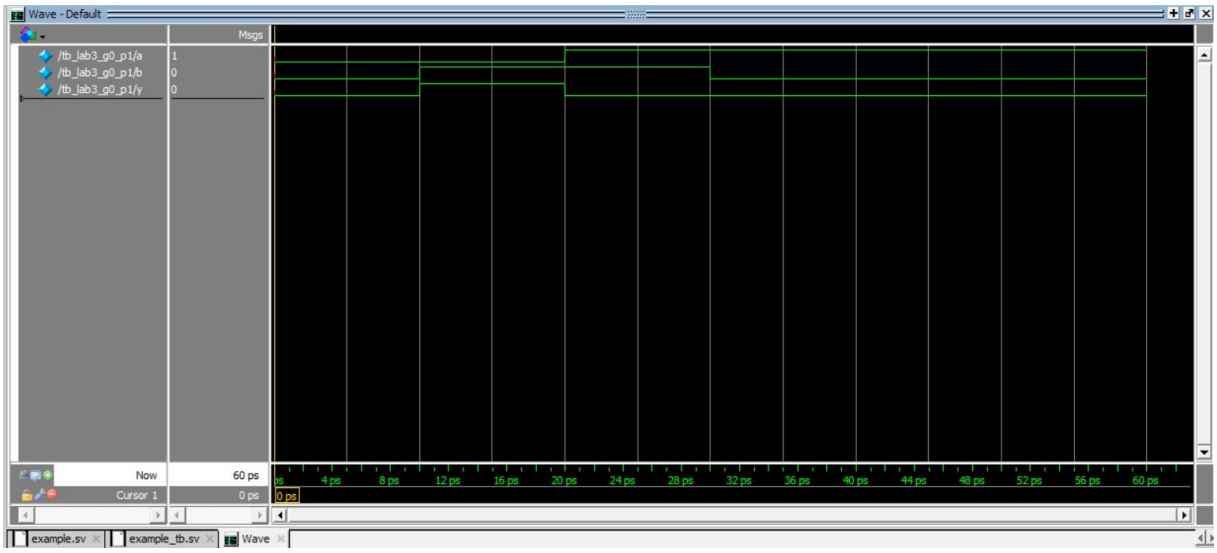
Şekil A ModelSim Programında Kodların Görüntüsü

```
module lab3_g0_p1
(    input  logic a, b,
  output logic y
);
  assign y = ~a & b;
endmodule
```

```
`timescale 1ns/1psmodule
tb_lab3_g0_p1 ,
logic a, b;
logic y;
lab3_g0_p1 uut0(.a(a), .b(b), .y(y));
a = 0; b = 0; #10;
b = 1;        #10;
a = 1;        #10;
b = 0;        #10;
              #20;
$stop;
end
endmodule
```

Tablo 1 ve Tablo 2 : Kodların İstenen Şekilde Rapora Eklenmesi

- Deneyde verilen Y= ~AB denklemini modelsim programını kullanarak tasarlanması istenmiştir. Modelsim programı açılarak File-> New-> Project seçilerek yeni bir proje oluşturulmuştur. Bu kısımda Teams dosyalarına eklenen videodan yardım alınmıştır. Oluşturulan projeye iki tane dosya eklenmiştir. Dosyalardan biri verilen denklemi oluşturmak için diğeri ise simülasyonda test etmek içindir. Dosya uzantılarına ve dosya isimlerine dikkat edilmiştir. Kod yazılıp derlendikten sonra 'Simulate ' bölümüne gelerek yazılan kod simule edilmiştir. Simulasyon çıktısı aşağıdaki gibi görülmüştür.



Şekil B Problem 1 Simülasyon Çıktısı

- **Çıkarım:** Simulasyon çıktısında görüldüğü gibi kurduğumuz devre beklenen şekilde çalışmaktadır. Devre sadece b=1 ve a=0 da 1 çıktısı vermektedir. And işlemi sadece iki durumunda 1 olduğu zaman çıktı olarak 1 verir. A'nın değili ifadesinden kaynaklı a=0 olduğu zaman ve b=1 olduğu zaman çıktı 1 olur. Hiçbir hata ve gecikme bulunmamaktadır.

2.2 Basit Bir Devrede Gecikme Simülasyonu

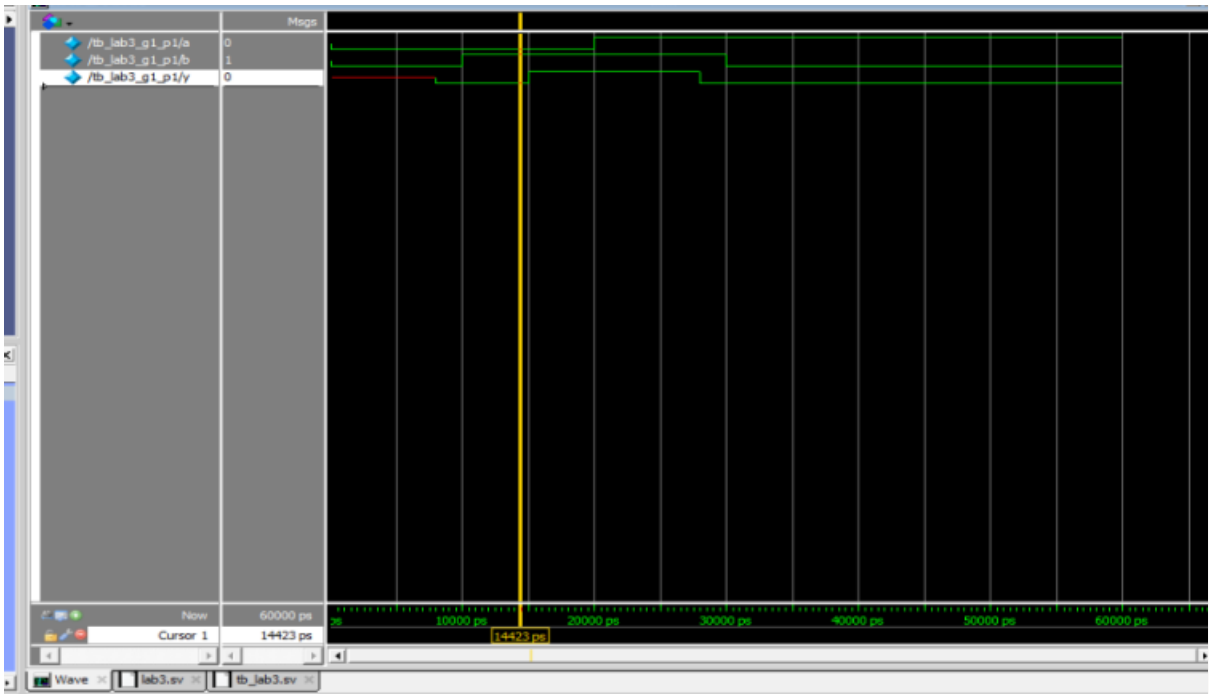
Problem1' de verilen koda ek olarak koda gecikmeler eklenmiştir. Verilog donanım dilinde bir kapıyı belirli bir zaman gecikmeli isteniyorsa '#..ns ' kodu yazılır. Problem1'de hazırlanan kodda 'not' kapısı ayrılarak n1 değişkenine atanmış ve 3 nano saniye gecikme eklenmiştir. 'Ve' kapısına ise #5 nano saniyelik bir gecikme eklenmiştir.2. kodun 1. Koddan bir diğer farkı da modül oluşturulan dosyada gecikme olduğu için timescale kodu testbenchde olduğu gibi bu dosyaya da eklenmiştir.

```
`timescale 1ns/1ps //delay için
module lab3_2 (
input logic a, b,
output logic y
);
assign #3 n1=~a; // wait 3 ns
assign #5 y = n1 & b; // wait 5 ns
endmodule
```

```
`timescale 1ns/1ps // testbench
module tb_lab3_2 ();
logic a, b; // all the inputs
logic y; // all the outputs
lab3_2 uut0(.a(a), .b(b), .y(y));
initial begin
a = 0; b = 0; #10; // a = 0, b = 0, wait 10 ns
b = 1; #10; // a = 0, b = 1, wait 10 ns
a = 1; #10; // a = 1, b = 1, wait 10 ns
b = 0; #10; // a = 1, b = 0, wait 10 ns
#20; // wait 20 ns after completion
$stop; // stop the simulation
end
endmodule
```

Tablo 3 ve Tablo 4 : Kodların İstenen Şekilde Rapora Eklenmesi

- Tablo 3 ve Tablo 4'te yazan kodlar simüle edildiğinde aşağıdaki dalga görüntüsü elde edilmiştir.



Şekil C Problem 2 Çıktısı

- **Çıkarım:** Ekran çıktısında da görüldüğü gibi çıkış değeri hem geç başlıyor hem de B inputu değişmesine rağmen output verilen gecikmeden dolayı geç değişiyor. Şekil b'de gösterilen 1. Probleme ait ekran çıktısında ise herhangi bir gecikme olmadığı için inputlar değiştiği anda output değişmekte idi. Fakat yukarıdaki çıktıda output sarı çizginin olduğu bölgede çoktan değişmesi gerekirken gecikmeli bir şekilde değişmiştir. Gecikmelerin eklendiği kodlar üzerinde değişimleri gözlemlemek mümkündür.

2.3 Glitch Simülasyonu

"Glitch" olarak adlandırılan istenmeyen geçişler, sinyalin bir devrede iki veya daha fazla yol alması ve bir yolun diğerinden daha uzun gecikmesi olması koşuluyla, bir giriş sinyali durum değiştirdiğinde ortaya çıkar.

A B \ C D	00	01	11	10
00	0	1	0	0
01	0	1	1	1
11	0	0	1	1
10	0	0	0	0

A B \ C D	00	01	11	10
00	0	1	0	0
01	1	1	1	1
11	0	0	1	0
10	0	0	0	0

A B \ C D	00	01	11	10
00	1	0	0	0
01	1	0	1	0
11	0	0	1	0
10	0	0	0	0

Şekil 3. K-haritası 1: Olası aksaklık, K-haritası 2: Aksaklık yok, K-haritası 3: Aksaklık yok.

$$X = A\,\overline{B}\,C + \overline{C}\,D$$

Denklem 2

- Öncelikle verilen devrenin K-Map haritasını çıkaralım.
- 4 girişli bir denklemden bahsettiğimiz için 2^4 = 16.

CD \ AB	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	1	0	0
10	0	1	0	0

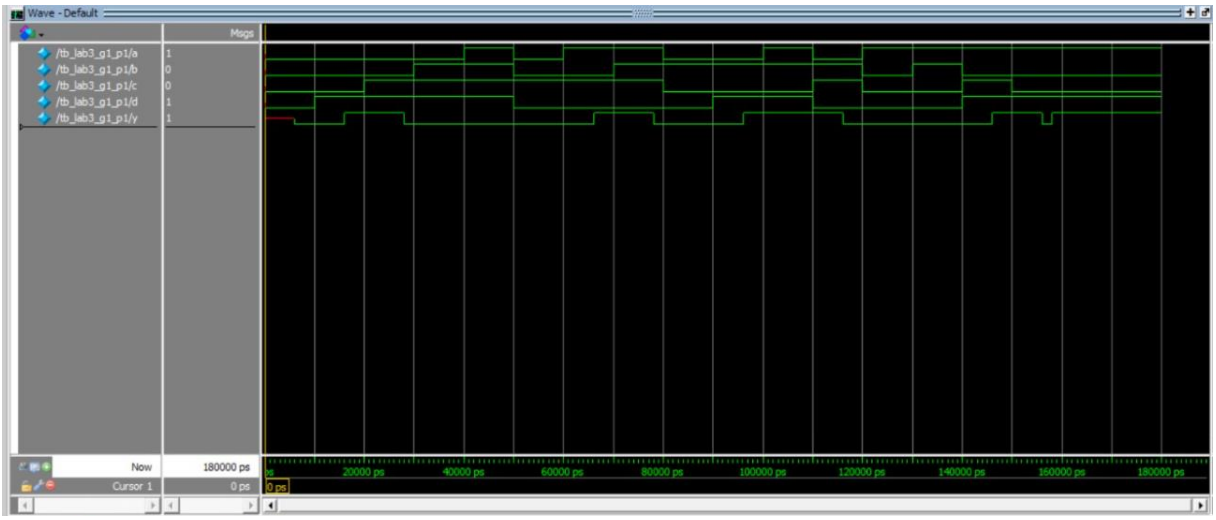
TABLO 6 : Verilen Denkleme Ait K-map Tablosu

- K-map de görüldüğü gibi sarı ile yeşil birlerin kesiştiği yerde glitch gözükmektedir. 0101'den 0111'e geçerken yani sadece C değiştiğinde devrede glitch gözlemlenmektedir.

```
`timescale 1ns/1ps
module lab3_g1_p1 (
input logic a, b,
 input logic c,d,
output logic y
);
logic n1,n2,n3,n4,n5;
assign #2 n1= ~b ;
assign #2 n2 =a & n1 & c ;
assign #2 n3= ~c ;
assign #2 n4= n3 & d;
 assign #2 n5=n4 | n2 ;
assign #2 y=n5 ;
endmodule
```

```
`timescale 1ns/1ps //testbench
module tb_lab3_g1_p1 ();
logic a, b,c,d; // all the inputs
logic y; // all the outputs
lab3_g1_p1 uut0(.a(a), .b(b),.c(c),.d(d),
.y(y));
initial begin
a=0;b=0;c=0;d=0; #10;
d=1; #10;
c=1;#10;
b=1; #10;
a=1; #10;
a=0; b=0;d=0; #10; a=1; #10;
b=1; #10;
a=0; b=1;c=0;d=0; #10;
d=1; #10;
a=1; #10;
a=0; c=1; d=0; #10;
a=1;b=0; c=0; #10;
b=1; #10;
b=0;d=1; c=1; #10;
c=0; #10;
#20; // wait 20 ns after completion
$stop; // stop the simulation
end
endmodule
```

Tablo 7 ve Tablo 8 : Kodların İstenen Şekilde Rapora Eklenmesi



Şekil D Problem 3 çıktısı

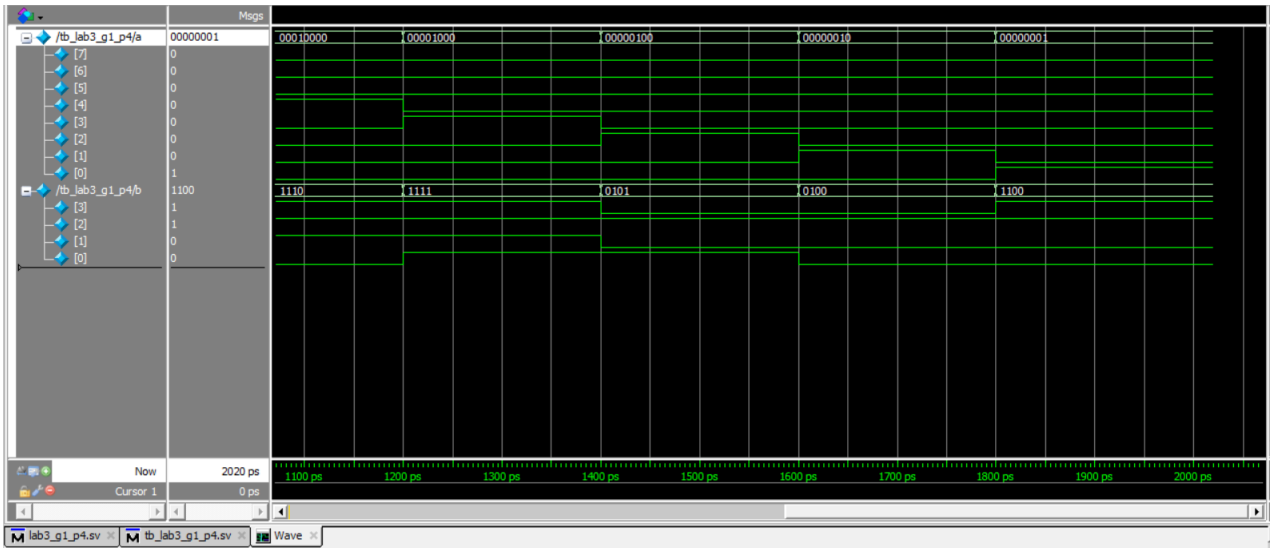
- Uygun giriş kombinasyonları testbench' de girildiğinde en sonda çıkış sinyalinin çok kısa bir zaman için 0'a düştüğü görülmektedir. Bu glitch'i ortadan kaldırmak için ise (A' & B & D) denklemi devreye eklenmelidir. Bunun sebebi birlerin çakıştığı yerde circle elde edilmesi ile oluşan glitch yapısının düzelmesidir.

2.4 Çözücü Tasarımı

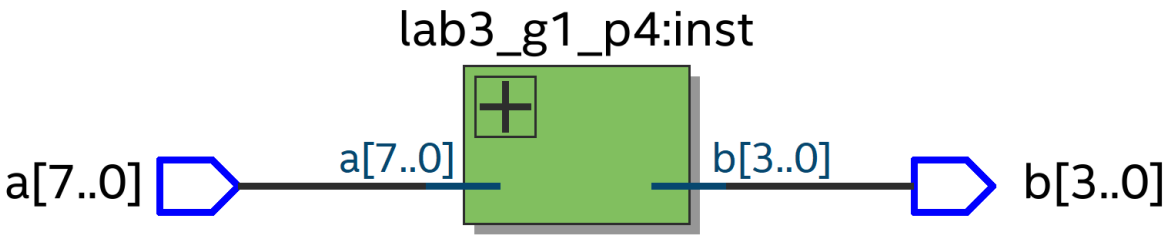
```
      module lab3_g1_p4 (
      input logic [7:0] a,
      output logic [3:0] b);

      (a[6]==1)?0011:
      (a[5]==1)?0010:
      (a[4]==1)?0110:
      (a[3]==1)?0111:
      (a[2]==1)?0101:
      (a[1]==1)?0100:
      (a[0]==1)?1100:
       0000);
endmodule
```

```
      module tb_lab3_g1_p4(); // testbench
logic [7:0] a;
      logic [3:0] b;
      lab3_g1_p4 uut0(.a(a), .b(b));
initial begin
      a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #200
      a[7]=1;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #200
      a[7]=1;a[6]=1;a[5]=1;a[4]=1;a[3]=1;a[2]=1;a[1]=1;a[0]=1; #200
      a[7]=0;a[6]=1;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #200
      a[7]=0;a[6]=0;a[5]=1;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #200
      a[7]=0;a[6]=0;a[5]=0;a[4]=1;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #200
      a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=1;a[2]=0;a[1]=0;a[0]=0; #200
      a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=1;a[1]=0;a[0]=0; #200
      a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=1;a[0]=0; #200
      a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=1; #200
#20; // wait 20 ns after completion
      $stop; // stop the simulation

      end
endmodule
```

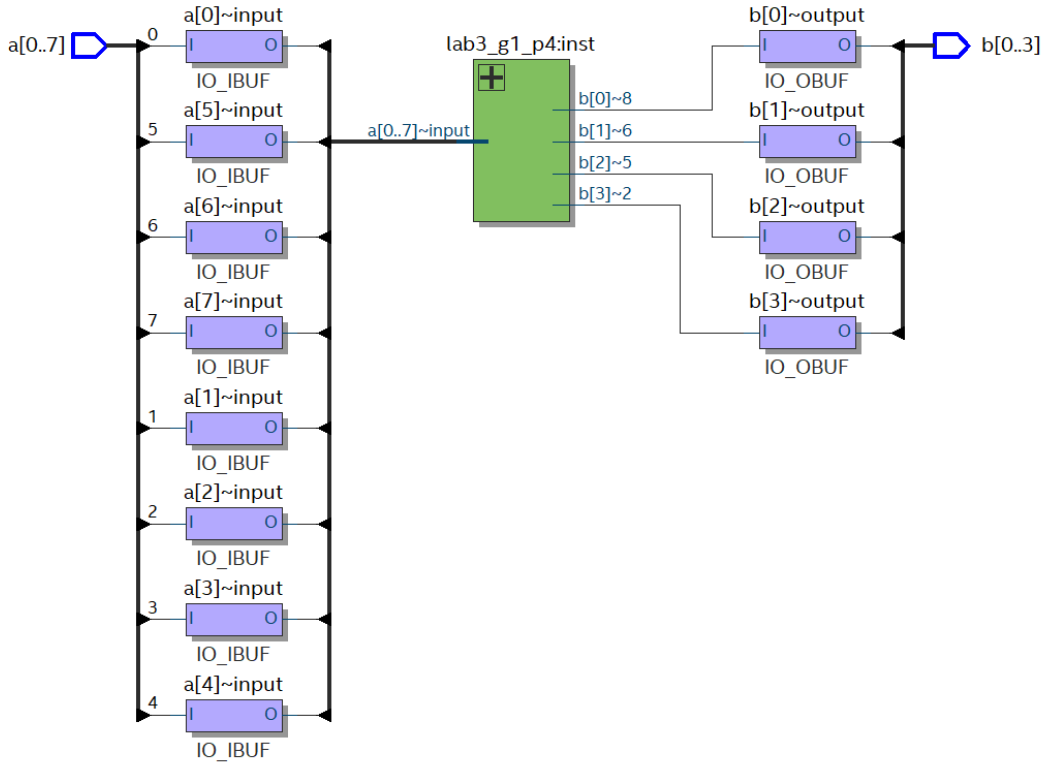
Tablo 9 ve Tablo 10 : Kodların İstenen Şekilde Rapora Eklenmesi



Şekil E Problem 4 çıktısı



Şekil F Problem 4 RTL çıktısı



Şekil G Problem 4 Post Fitting çıktısı

Analysis & Synthesis Resource Usage Summary

<<Filter>>

	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	3
2	-- 3 input functions	5
3	-- <=2 input functions	1
5		
6	Logic elements by mode	
1	-- normal mode	9
2	-- arithmetic mode	0
7		
8	Total registers	0
1	-- Dedicated logic registers	0
2	-- I/O registers	0
9		
10	I/O pins	12
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	a[7]~input
15	Maximum fan-out	4
16	Total fan-out	45
17	Average fan-out	1.36

Fitter Status	Successful - Sat Apr 04 20:26:18 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	lab3
Top-level Entity Name	lab3
Family	MAX 10
Device	10M50DAF484I7G
Timing Models	Final
Total logic elements	9 / 49,760 (< 1 %)
Total registers	0
Total pins	12 / 360 (3 %)
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 %)

- Verilog dili birkaç kodlama stilinde bir modül tasarlama yeteneğine sahiptir.
- 1- Davranışsal veya Algoritmik seviye (Sentezlenmesi en zor olan)
- 2- Veri Akışı Düzeyi
- 3-Kapı seviyesi veya Yapısal Seviye
- 4- Switch Level

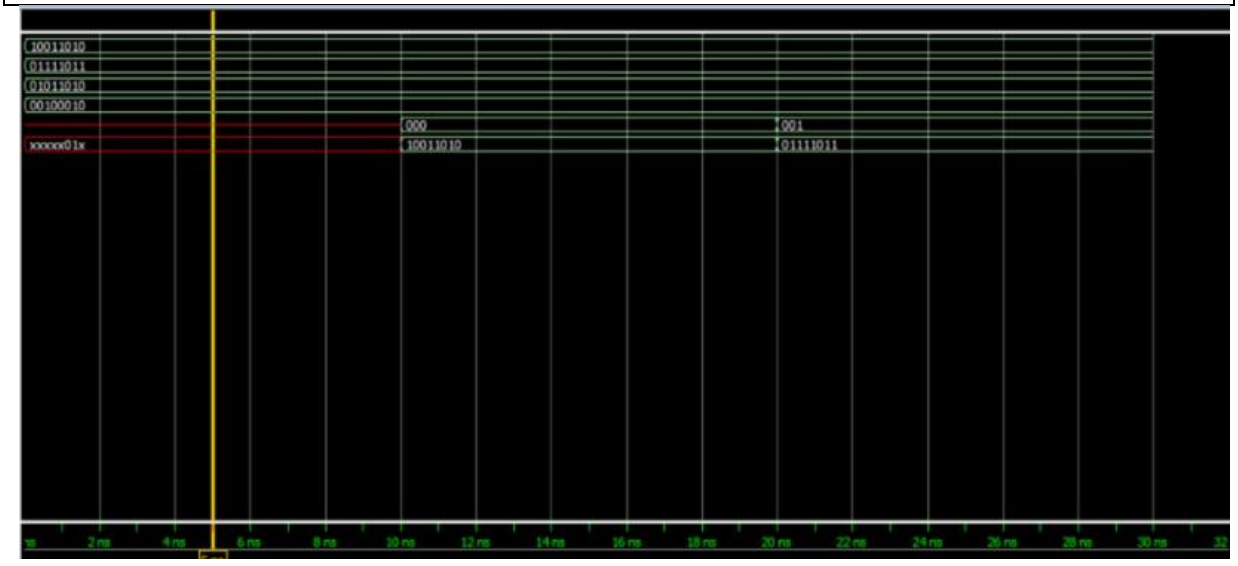
Yukarıda belirtilen sıra, en yüksekten en düşük soyutlama düzeyine doğrudur. Eğer devre daha yüksek seviye soyutlama yapılıp öyle kurulmuş olsa idi,hem kod yazma kısmı hemde kodu anlama kısmı daha kolay olurdu fakat devre de bazı önemli detaylar kaybolabilir idi. Yani High Abstraction yapmak daha az detay görülmesini sağlar, Low Abstraction yapmak ise daha fazla detay görülmesini sağlar. High Abstraction yapılmış bir devrede oluşan sorunları görmek zordur, çünkü detay azdır.

- Devre hafızada 49.760 bits yer kaplamaktadır.

2.5) Yapısal Tasarım (Structural Design)

```
module mux4(
    input logic [7:0] a,b,c,d,
    input logic [2:0]  secim,
    output logic [7:0] output3
);

wire [7:0] output1,output2;
mux2 #(8) ilkmux(a,b,secim[0],output1);
mux2 #(8) ikincimux(c,d,secim[1],output2);
mux2 #(8) ucuncumux(output1,output2,secim[2],output3);

endmodule


module mux2  #(parameter WIDTH = 8 )                //
              (input logic[WIDTH-1:0]   d0,d1,      //
               input logic              s,          //
               output logic[WIDTH-1:0]  y      );   //

assign y = s ? d1 : d0 ;                            //
endmodule
```

```
`timescale 1ns/1ns

module tb_mux4();

 logic [7:0] a,b,c,d;

logic [2:0]  secim;

logic [7:0] output3;

mux4 deneme( .a(a), .b(b), .c(c), .d(d), .secim(secim),
.output3(output3)  );


 initial begin

 a= 154;
 b= 123;
 c= 90;
 d=34;
#10;
 secim= 3'b000;
#10;
  secim= 3'b001;
#10;
 $stop;
 end
endmodule
```



Şekil H Problem 5 ModelSim çıktısı

- **Yapısal Tasarımın Avantajları:**

- Modül, mantık kapıları ve bu kapılar arasındaki ara bağlantılar açısından uygulanmaktadır.Sinyallerle bağlantılı bileşenlerle şematik bir çizime benzer.

- Anahtar düzeyinde soyutlama nadiren kullanıldığından, kapı düzeyinde modelleme neredeyse en düşük soyutlama düzeyidir. Kapı seviyesi modelleme, çoklayıcılar, tam toplayıcı, vb. gibi bir tasarımdaki en düşük seviyeli modülleri uygulamak için kullanılır. Verilog'un tüm temel kapılar için kapı ilkelleri vardır.
  Sadece **kombinasyonel devreler için kullanılır.**

- **Giriş değiştikçe** anında değer güncellemesi

- İki veya daha fazla girdi aynı anda değişiyorsa, eylemlerini **aynı anda gerçekleştireceklerdir.**

- Bu soyutlama seviyesinde çok sayıda ayrıntı vardır, ancak yine **de gerçek fiziksel uygulamaya daha yakındır** , bu yeni gelenlerin kodu devre tasarımı ile ilişkilendirmesini kolaylaştırır.

  Referanslar:
https://www.quora.com/What-is-GATE-level-modelling-in-Verilog
https://learn.digilentinc.com/Documents/277
https://www.youtube.com/watch?v=d8zmZVHLufM