



GEBZE TEKNİK ÜNİVERSİTESİ
ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 3 Deney Raporu

Donanım Tanımlama Dillerine Giriş

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

1. Teorik Araştırma

Verilog elektronik sistemleri modellemek için kullanılan bir donanım tanımlama dilidir. Verilog (bazen “Verilog HDL” olarak da adlandırılır) analog, sayısal ve karışık işaretli devrelerin tasarımını, doğrulanmasını ve yürütülmesini değişik düzeylerde desteklemektedir. Verilog dilinin tasarımcıları dilin C programlama diline yakın bir söz dizimine sahip olmasını istemişlerdir. Böylece bu dile yatkın olan mühendislerin dili kolayca kullanmasını amaçlamışlardır. Dil küçük/büyük harf duyarlılığına sahiptir ve temel denetim akışının “if” ve “while” gibi anahtar kelimeleri, C'ye benzemektedir. Verilog birkaç temel yönde C'den farklıdır. Verilog bir blok kodu tanımlamak için kıvrık parantezler yerine Begin/End kullanmaktadır. Verilog 95 ve 2001 işaretçi veya yinelemeli alt yordamlar yapılarına sahip değildir fakat SystemVerilog bu özelliklere sahiptir.

Frank Gray'den sonra yansıtılan ikili (RB) veya Gray kodu olarak da bilinen yansıyan ikili kod (RBC), iki ardışık değerin yalnızca bir bitte (ikili basamak) farklı olacağı şekilde ikili sayı sisteminin bir sıralamasıdır .Örneğin, "1" ondalık değerinin ikili sistemde temsili normalde " 001 " ve "2", " 010 " olacaktır. Gray kodunda bu değerler " 001 " ve " 011 " olarak gösterilir. Bu şekilde, 1'den 2'ye bir değeri artırmak, iki yerine yalnızca bir bitin değişmesini gerektirir. Gri kodlar, elektromekanik anahtarlardan gelen sahte çıkışı önlemek ve dijital karasal televizyon ve bazı kablolu TV sistemleri gibi dijital iletişimde hata düzeltmeyi kolaylaştırmak için yaygın olarak kullanılmaktadır.

Nesne yönelimli programlamada Soyutlama (Abstraction) ilkesi, eğer bir sınıf için nesne üretmek mantıksız geliyorsa o sınıf soyutlanabilir. Alt sınıfların ortak özelliklerini ve işlevlerini taşıyan ancak henüz bir nesnesi olmayan bir üst sınıf oluşturmak istenirse bir soyut (abstract) üst sınıf oluşturulur. Soyutlama, bir sınıfa veya metoda temel görevlerin tanımlanması, detayların ise tanımlanmaması demektir. Temel olarak bir soruna ait çözüme giderken kullanılacak yöntemlerin, ilk etapta daha genel basit ve soyut bir tanımını yapmaktır.

Decimal	Binary	Gray	Decimal of Gray
0	0000	0000	0
1	0001	0001	1
2	0010	0011	3
3	0011	0010	2
4	0100	0110	6
5	0101	0111	7
6	0110	0101	5
7	0111	0100	4
8	1000	1100	12
9	1001	1101	13
10	1010	1111	15
11	1011	1110	14
12	1100	1010	10
13	1101	1011	11
14	1110	1001	9
15	1111	1000	8

2. Problemler

2.1 Problem 1 – Basit bir devre tasarımı ve simülasyonu

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

Denklem 1

A. Denklem 1 deki Boolean denklemini HDL ile tasarlayın.

B. Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemleyin.

C. ModelSim dalga şemasını File→Export→Image.. ile PNG olarak çıkarabilirsiniz. Veya tüm ModelSim projesinin ekran görüntüsü alabilirsiniz.

2.1.1 Problemin çözümü

A.

```

C:/modelsimlab/lab3_g15_p1.sv - Default
Ln#
1  /* lab3_g15_p1.sv
2  *
3  * Hazırlayanlar:
4  * Senanur Ağaç
5  * Beyza Duran
6  * Notlar:
7  * ELM235 2020 Bahar Lab3 - Problem 1
8  * Y = NOT A and B denkleminin gerçekleştirilmesi
9  *
10 /*
11 module lab3_g15_p1 (
12     input logic a, b,
13     output logic y
14 );
15     assign y = ~a & b;
16 endmodule
17

```

Şekil 1:HDL tasarımı

B.

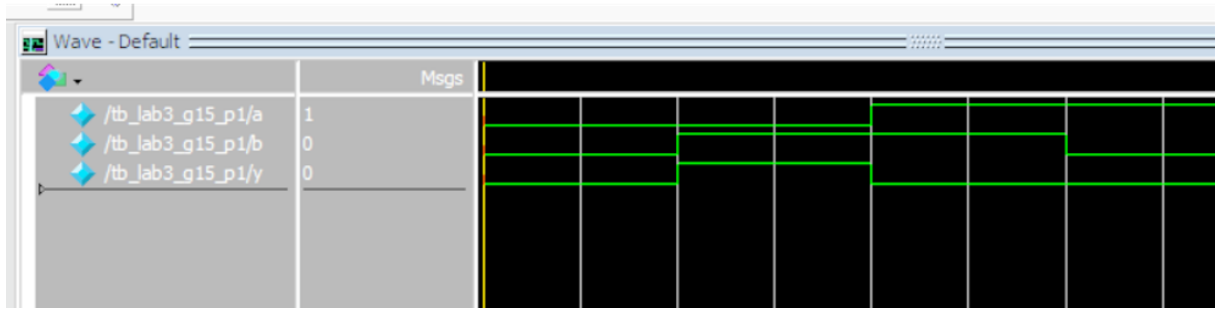
```

C:/modelsimlab/tb_lab3_g15_p1.sv - Default
Ln#
1  /* lab3_g15_p1.sv
2  *
3  * Hazırlayanlar:
4  * Senanur Ağaç
5  * Beyza Duran
6  * Notlar:
7  * ELM235 2020 Bahar Lab3 - Problem 1
8  * Y = NOT A and B denkleminin gerçekleştirilmesi
9  *
10 /*
11 module tb_lab3_g15_p1 ();
12 // These names do not need to be the same as the part1 ports
13 // but, we make them the same to make it easier to understand
14 // which is connected to what port
15     logic a, b; // all the inputs
16     logic y; // all the outputs
17 // Explicit port mapping. Always prefer it, vs. the implicit.
18     lab3_g15_p1 uut0(.a(a), .b(b), .y(y));
19 // This part is applied to the circuit sequentially.
20 // The results can be inspected
21     initial begin
22         a = 0; b = 0; #10; // a = 0, b = 0, wait 10 ns
23         b = 1; #10; // a = 0, b = 1, wait 10 ns
24         a = 1; #10; // a = 1, b = 1, wait 10 ns
25         b = 0; #10; // a = 1, b = 0, wait 10 ns
26         #20; // wait 20 ns after completion
27         $stop; // stop the simulation
28     end
29 endmodule

```

Şekil 2: Testbench kodları

C.



Şekil 3: Dalga Şeması

2.2 Problem 2 - Basit bir devrede gecikme simülasyonu

A. Problem 1 deki devreleriniz kopyalayın, NOT kapısı sonucu için dahili bir sinyal oluşturun, ve NOT kapısını 3ns, AND kapısını 5 ns gecikme ile atayın. Bunun için devrenin başına timescale eklemeniz gerekmektedir. Bunun için her bir kapıyı ayrı parçalara bölmeniz gerekmektedir. Aşağıda bir örneği verilmiştir.

assign #3 n1 = ~a;

assign #5 y = n1 and b;

B. Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemleyin.

C. Çıkan dalga şemasını yorumlayın.

2.2.1 Problemin Çözümü

A.

```

C:/modelsim/SORU2/lab3_g15_p2.sv (/tb_lab3_g15_p2/uut0) - Default
Ln#
1      `timescale 1ns/1ps
2      module lab3_g15_p2 (
3      input logic a, b,
4      output logic y
5      );
6      //assign y = ~a & b;
7      assign #3 n1 = ~a;
8      assign #5 y = n1 & b;
9      endmodule
10

```

Şekil 4: Devre tasarımı

B.

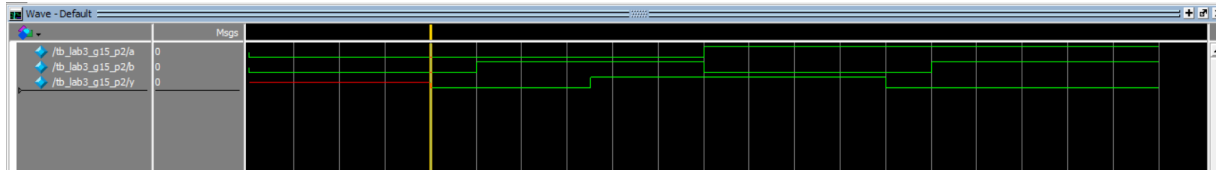
```

C:/modelsim/SORU2/tb_lab3_g15_p2.sv (/tb_lab3_g15_p2) - Default
Ln#
1
2 `timescale 1ns/1ps
3 module tb_lab3_g15_p2 ();
4 // These Names do not need to be the same as the part1 ports
5 // but, we make them the same to make it easier to understand
6 // which is connected to what port
7 logic a, b; // all the inputs
8 logic y; // all the outputs
9 // Explicit port mapping. Always prefer it, vs. the implicit.
10 lab3_g15_p2 uut0(.a(a), .b(b), .y(y));
11 // This part is applied to the circuit sequentially.
12 // The results can be inspected
13 initial begin
14 a = 0; b = 0; #10; // a = 0, b = 0, wait 10 ns
15 a = 0; b = 1; #10; // a = 0, b = 1, wait 10 ns
16 a = 1; b = 0; #10; // a = 1, b = 0, wait 10 ns
17 a = 1; b = 1; #10; // a = 1, b = 1, wait 10 ns
18
19 $stop; // stop the simulation
20 end
21 endmodule
22

```

Şekil 5: Testbench çıktısı

C.



Şekil 6 : Dalga Şeması

Problem 1’de ki oluşturulan devreye göre dalga şemasında gecikme gözlenmektedir. Kapılara eklenen gecikme süreleri ile dalga şemasında kayma meydana gelmiştir.

2.3 Problem 3 - Glitch simülasyonu

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

Denklem 2

A. Denklem 2 deki Boolean denklemini HDL kullanarak tasarlayın. Her bir kapıya 2ns gecikme vererek atama yapın (NOT, AND ve OR)

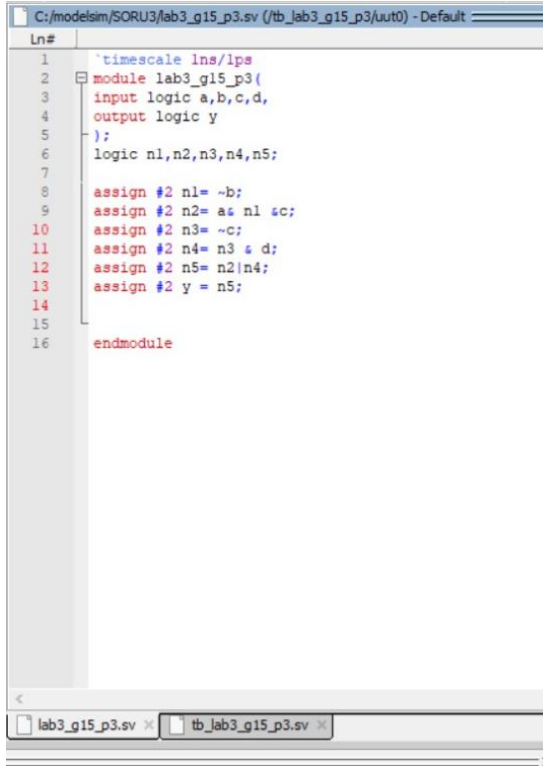
B. Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemleyin. Her bir giriş ataması arasında 10ns bekleyin. Burada glitch oluşup oluşmadığı hakkında bilgi verin.

C. Devrenin K-Mapine bakarak, hangi koşulda glitch oluşabileceği hakkında yorum yapın, ve bulduğunuzu o koşulu Testbench te test edin.

D. Çıkan dalga şeklini yorumlayın, ve glitchi ortadan kaldırmak için nasıl bir devre eklenmesi gerektiğini belirtin.

2.3.1 Problemin Çözümü

A.



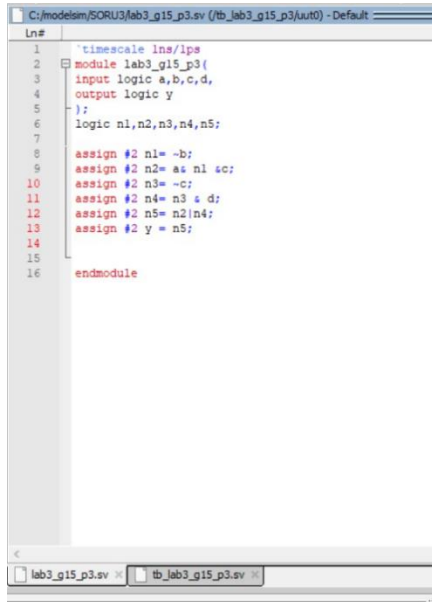
```

C:/modelsim/SORU3/lab3_g15_p3.sv (/tb_lab3_g15_p3/aut0) - Default
Ln#
1 `timescale 1ns/1ps
2 module lab3_g15_p3(
3   input logic a,b,c,d,
4   output logic y
5 );
6   logic n1,n2,n3,n4,n5;
7
8   assign #2 n1= ~b;
9   assign #2 n2= a&n1 &c;
10  assign #2 n3= ~c;
11  assign #2 n4= n3 & d;
12  assign #2 n5= n2|n4;
13  assign #2 y = n5;
14
15
16 endmodule
  
```

Şekil 7: HDL devre tasarımı

Denklem 2 deki Boolean denklemi HDL kullanarak tasarlanmıştır. Her bir kapıya 2ns gecikme vererek atama yapılmıştır

B.

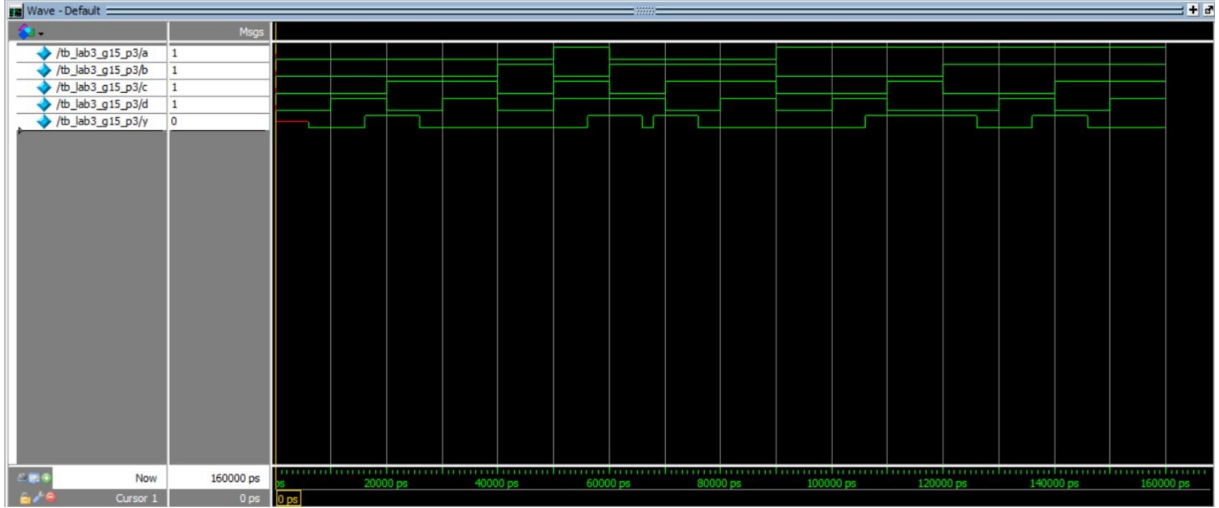


```

C:/modelsim/SORU3/lab3_g15_p3.sv (/tb_lab3_g15_p3/aut0) - Default
Ln#
1 `timescale 1ns/1ps
2 module lab3_g15_p3(
3   input logic a,b,c,d,
4   output logic y
5 );
6   logic n1,n2,n3,n4,n5;
7
8   assign #2 n1= ~b;
9   assign #2 n2= a&n1 &c;
10  assign #2 n3= ~c;
11  assign #2 n4= n3 & d;
12  assign #2 n5= n2|n4;
13  assign #2 y = n5;
14
15
16 endmodule
  
```

Testbench oluşturularak, devrenin bütün girişlere karşı nasıl davrandığını gözlemlenmiştir. Her bir giriş ataması arasında 10ns beklenip şemalar oluşturulmuştur.

Şekil 8: Testbench çıktısı



Şekil 9 : Dalga Şeması

C. Glitch, sistemde kısa ömürlü bir arızadır. Genellikle kendini düzelten geçici bir hatayı tanımlamak için kullanılır ve bu nedenle sorunu gidermek zordur. 4 girişli denkleminizin Kmap tablosunun 5. ve 6. satırlarında ani bir şekilde 1'den 0'a ve 0'dan 1'e geçişler yaşanmış ve bu da dalga şemasında glitch (kusur) olayına neden olmuştur. Bu olay dalga şemasında da görülmektedir.

D.

CD	0 0	0 1	1 1	1 0
AB				
0 0	0	1	0	0
0 1	0	1	0	0
1 1	0	1	0	0
1 0	0	1	1	1

Tablo1'de belirtilen şekildeki Kmap tablosundaki gibi seçildiğinde glitch oluşmaz. $C'D + AB'$ Şeklinde denklem oluşur.

Tablo1 :Kmap tablosu

CD	0	0	1	1
AB				
0 0	0	1	0	0
0 1	0	1	0	0
1 1	0	1	0	0
1 0	0	1	1	1

Tablo2 :Kmap tablosu

CD \ AB	0 0	0 1	1 1	1 0
0 0	0	1	0	0
0 1	0	1	0	0
1 1	0	1	0	0
1 0	0	1	1	1

$$A'C'D + C'DA + CAB'$$

2. ve 3. tabloda glich oluşmamsı adına Kmap tablosunda hiçbir kesişim noktası seçilmeden 1'ler dahil edilmiştir.

2.4 Problem 4 - Çözücü tasarımı

A. 8-girişli bir devrenin, verilen isterlere uygun girişlerinin önceliğini belirleyen bir devre tasarlayınız. Girişiniz 8-bitlik bir bus olsun. Çıkışınız 4-bitlik bir bus olsun. Girişin 8. elemanı (i.e. a[7]) aktif ise, çıkış 0001, 8. girişi aktif değil ve 7. girişi aktif ise (i.e. a[7]' a[6]) çıkış 0011, gibi çıkış sırası Grey Code ile kodlanmış bir devreyi sadece Şartlı Atamalar (Conditional Assignments) kullanarak tasarlayınız.

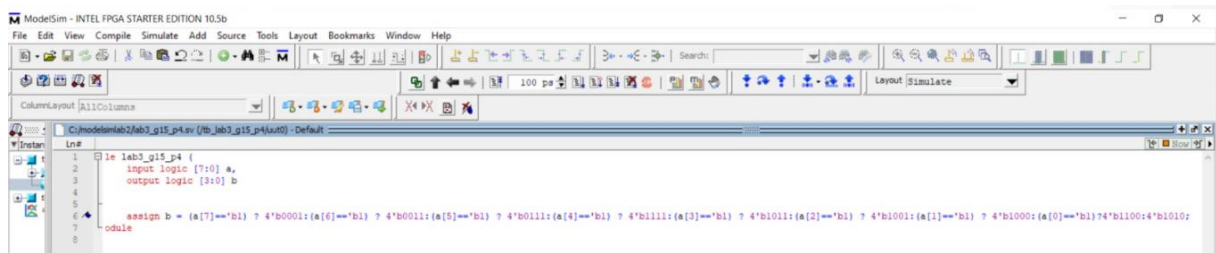
B. Devrenin simülasyonunu yaparak, rastgele seçtiğiniz 16 farklı giriş kombinasyonuna göre çıkış dalga şeklini gözlemleyiniz.

C. Devreyi Sentezleyerek RTL ve Post-Fitting şemalarını ekleyiniz, ne kadar yer kapladığı hakkında bilgi veriniz.

D. Devreyi oluşturmanın alternatif bir yolundan bahsederek, devreyi daha yüksek seviye soyutlamada tanımlamanın avantajları veya dezavantajları hakkında yorum yapınız.

2.4.1 Problemin Çözümü

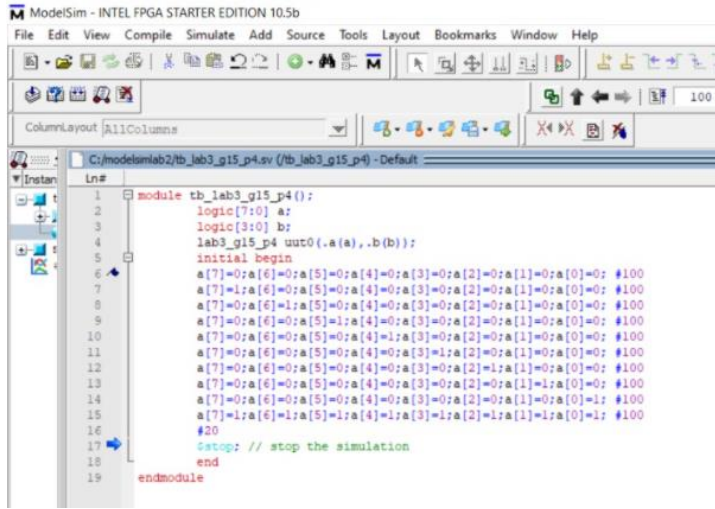
A.



Şekil 10 : HDL devre tasarımı

A şıkında verilen bilgiler doğrultusunda HDL kullanarak tasarlanmıştır. Conditional Assignments ile girişlerin her bir elemanına şartlı atama yapılmıştır.

B.



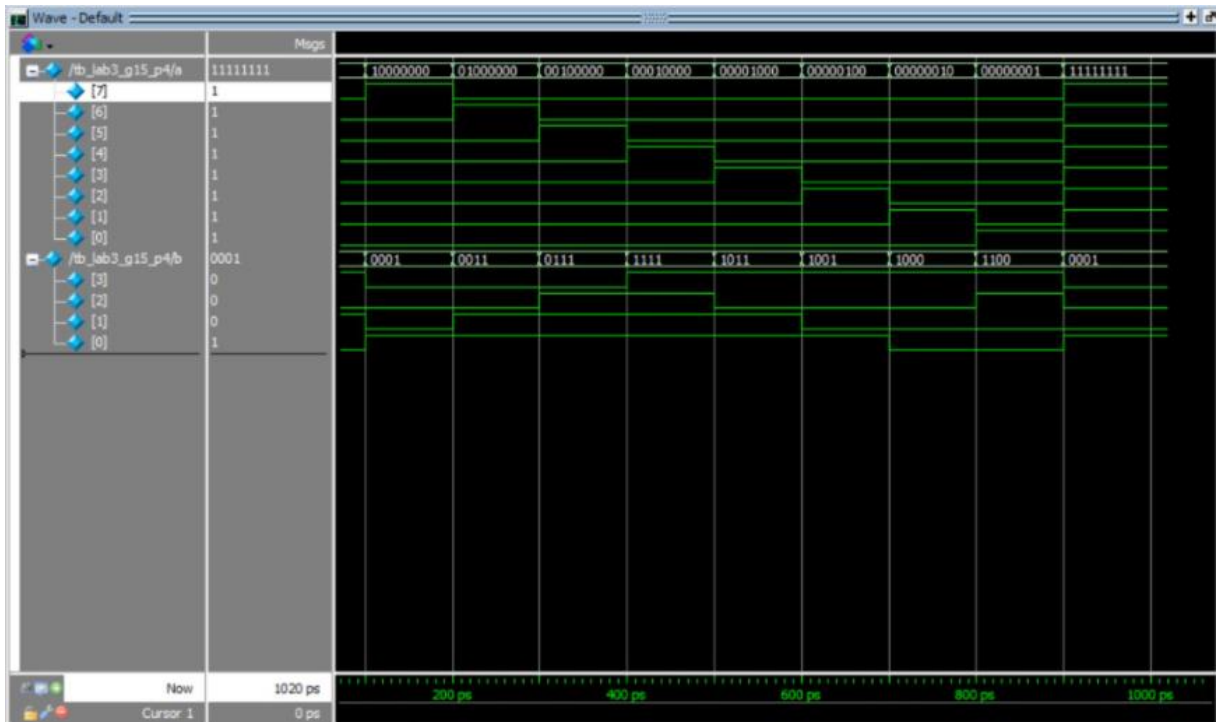
```

1 module tb_lab3_g15_p4();
2   logic[7:0] a;
3   logic[3:0] b;
4   lab3_g15_p4 uut0(.a(a),.b(b));
5   initial begin
6     a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #100
7     a[7]=1;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #100
8     a[7]=0;a[6]=1;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #100
9     a[7]=0;a[6]=0;a[5]=1;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #100
10    a[7]=0;a[6]=0;a[5]=0;a[4]=1;a[3]=0;a[2]=0;a[1]=0;a[0]=0; #100
11    a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=1;a[2]=0;a[1]=0;a[0]=0; #100
12    a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=1;a[1]=0;a[0]=0; #100
13    a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=1;a[0]=0; #100
14    a[7]=0;a[6]=0;a[5]=0;a[4]=0;a[3]=0;a[2]=0;a[1]=0;a[0]=1; #100
15    a[7]=1;a[6]=1;a[5]=1;a[4]=1;a[3]=1;a[2]=1;a[1]=1;a[0]=1; #100
16    #20
17    $stop; // stop the simulation
18  end
19 endmodule

```

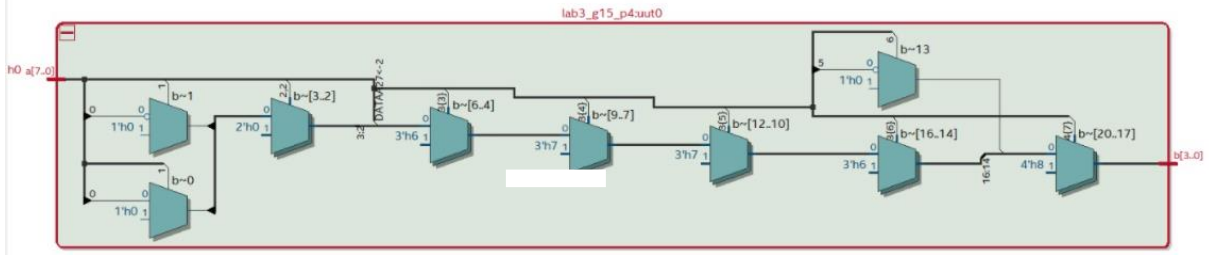
Tüm kombinasyonları deneyerek dalga şeması çıkarılır.

Şekil 11: Testbench çıktısı

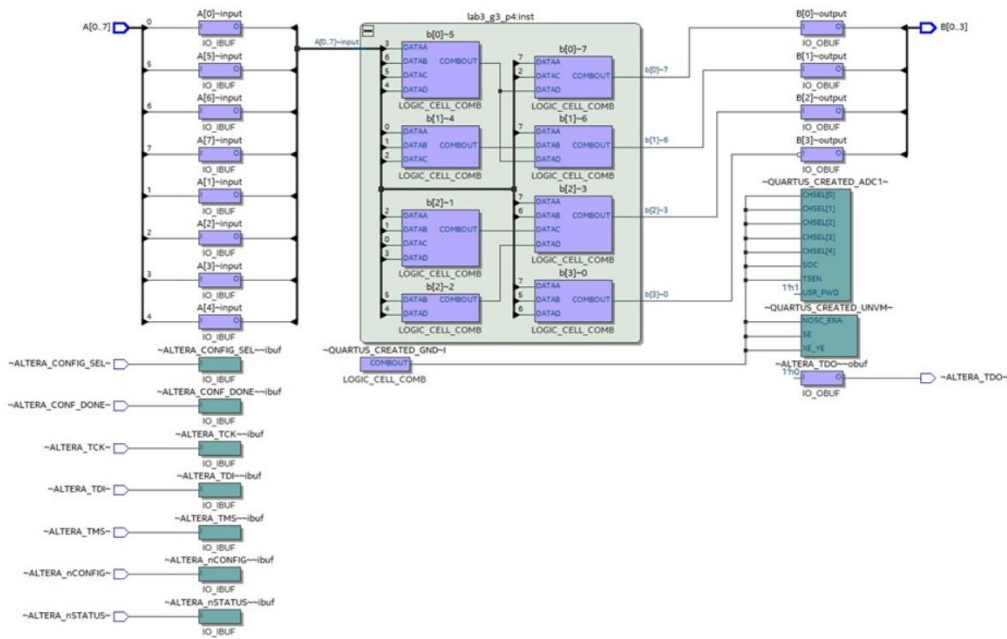


Şekil 12 : Dalga Şeması

C.



Şekil 13 : RTL devre sentezi



Şekil 14 : Post fitting

D.

Devreyi daha yüksek seviye soyutlama ile oluşturabiliriz.

Daha yüksek bir soyutlama ile sistem içindeki tüm üst düzey bileşenlerin davranışlarını ve etkileşimlerini daha iyi anlaşılabilir ve bu nedenle genel sistemin karmaşıklığına karşı koymak için devre daha donanımlı hale gelir.

2.5 Problem 5 - Yapısal tasarım (Structural Design)

- A. 8-bitlik 2-ye-1 seçici devresi tasarlayın.
- B. A daki devreyi kullanarak 8-bitlik 4-e-1 seçici devre tasarlayın.
- C. B deki devrenin simülasyonunu yaparak, doğruluğunu gözlemleyin.
- D. B deki devreyi Sentezleyerek RTL ve Post-Fitting şemalarını ekleyiniz, ne kadar yer kapladığı hakkında bilgi veriniz.
- E. Yapısal Tasarımın avantajları hakkında yorum yapınız.

2.5.1 Problem Çözümü

A.

```

C:/MODELSIM5/lab3_g15_p5.sv (/lab3_g15_p5) - Default *
Ln#
1 module lab3_g15_p5( input logic clk, s,
2                     input logic [7:0] a1,b1,
3                     output logic [7:0] x);
4
5
6 end
7 endmodule
8
9 module mux4(
10    input logic [7:0] a,b,c,d,
11    input logic [2:0] k,
12    output logic [7:0] o3
13 );
14
15 wire [7:0] o1,o2;
16
17 lab3_g15_p5 mux1(a,b,k[0],output1);
18
19 lab3_g15_p5 mux2(c,d,k[1],output2);
20
21 lab3_g15_p5 mux3(o1,o2,k[2],o3);
22
23 endmodule
24

```

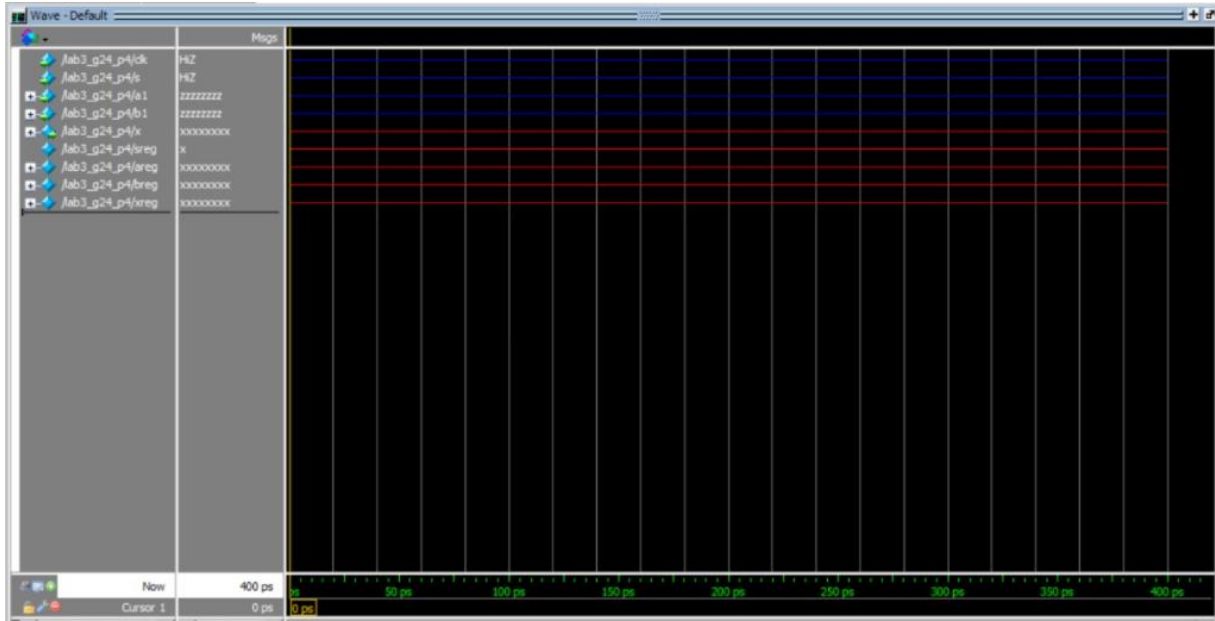
B.

```

C:/MODELSIM5/lab3_g15_p5.sv (/lab3_g15_p5) - Default *
Ln#
1 module lab3_g15_p5( input logic clk, s,
2                     input logic [7:0] a1,b1,
3                     output logic [7:0] x);
4
5
6 end
7 endmodule
8
9 module mux4(
10    input logic [7:0] a,b,c,d,
11    input logic [2:0] k,
12    output logic [7:0] o3
13    output logic [7:0] o4
14 );
15
16 wire [7:0] o1,o2;
17
18 lab3_g15_p5 mux1(a,b,k[0],output1);
19
20 lab3_g15_p5 mux2(c,d,k[1],output2);
21
22 lab3_g15_p5 mux3(o1,o2,k[2],o3);
23 lab3_g15_p5 mux3(o1,o2,k[2],o4);
24
25 endmodule
26

```

C.



E . Girişleri ve çıkışları olan bir donanım bloğuna modül denir. Bir AND geçidi, bir çoklayıcı ve bir öncelik devresinin tümü, donanım modüllerinin örnekleridir. Modül işlevselliğini tanımlayan iki genel stil, davranışsal ve yapısaldir. Davranışsal modeller, bir modülün ne yaptığını tanımlar. Yapısal modeller, bir modülün daha basit parçalardan nasıl oluşturulduğunu tanımlar; hiyerarşinin bir uygulamasıdır. Yapısal tasarım kullanışlı ucuz ve güvenlidir.

SONUÇLAR:

Öncelikle ModelSim programı üzerinden basit bir devrenin simülasyonu yapılmış ve dalga şeması çıkarılmıştır. Daha sonra bu devrede gecikme durumu için kapı eklenmiş ve kapılara 3ns ve 5ns olmak üzere gecikme süreleri atanmıştır. Çıkan dalga şeması ile ilk durumdaki şema karşılaştırılarak gecikme gözlenmiştir.

Verilen denkleme göre Kmap oluşturulmuş ve bu tabloda ani değişim görülen satırlara karşılık gelen dalga şemasında anlık kırılmalar oluştuğu görülmüştür. Burada glitch olayı gözlemlenmiştir.

Daha sonra 8 bit girişli ve 4 bitlik girişli bir devre tasarlayarak bu elemanlar şartlı atama ile istenilen çıkışlara atanmıştır. ModelSim üzerinden gerekli kodlar yazılmış olup QuatusPrime programına aktarılmış ve sentezlenen devre şeması ve Post Fitting şemaları eklenmiştir.

Son olarak 5. problemde devreler tasarlanmış fakat Quarus Prime programı üzerinden sentezlenen RTL şeması ve Post Fitting şemaları eklenememiştir.

KAYNAKÇA

https://en.wikipedia.org/wiki/Gray_code

<https://app.patika.dev/courses/oop/abstraction>

<https://tr.wikipedia.org/wiki/Verilog>