GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 2 Deney Raporu

Donanım Tanımlama Dilleri Uygulamaları: Birleşik Devreler

Hazırlayanlar
1) 200102002027 – Erhan Gök
2) 200102002028 – Esra Kirman

1. Giriş

Deneyde; donanım tanıma dillerini (DTD) kullanarak devre tasarımı yapılışı, sentezleyici araçları kullanarak DTD ile tanımlanan devreleri FPGA için sentezleme, simülasyon araçları kullanarak otomatik simülasyon yaptırma ve devrede oluşan gecikmeleri gözlemleme konularında öğrenim ve tecrübe kazanılması amaçlanmıştır.

2. Problemler

2.1. Problem I - İstenmeyen Darbe Sinyalleri: Glitchler

2.1.1. Teorik Araştırma

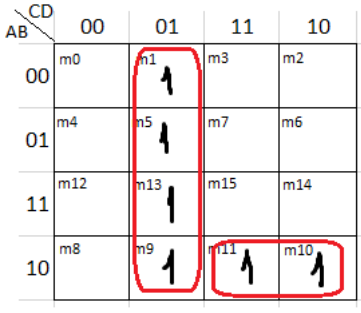
Propagasyon gecikmesi iletimde başlangıçtan hedefe ulaşana kadar geçen süreyi ifade eder. Bu gecikme bir mantık kapısına giriş kararlı ve değişmek için geçerli olduğunda başlayan, bu mantık kapısının çıkışının kararlı olduğu zamana kadar geçen sürenin uzunluğudur. [1] Bu gecikmeyi modellemek için her kapıya 2 ns gecikmeyi `assign #2` kodu ile atarız. DTD kodunun başına da `` `timescale 1ns/1ps `` komutunu koyarız.

Y = AB'C + C'D denkleminin doğruluk tablosu aşağıdaki gibi oluşur:

A	B	C	D	Y	
0	0	0	0	0	m0
0	0	0	1	1	m1
0	0	1	0	0	m2
0	0	1	1	0	m3
0	1	0	0	0	m4
0	1	0	1	1	m5
0	1	1	0	0	m6
0	1	1	1	0	m7
1	0	0	0	0	m8
1	0	0	1	1	m9
1	0	1	0	1	m10
1	0	1	1	1	m11
1	1	0	0	0	m12
1	1	0	1	1	m13
1	1	1	0	0	m14
1	1	1	1	0	m15

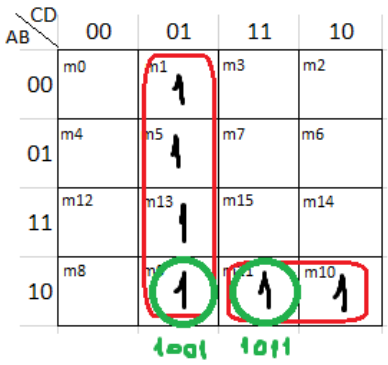
Şekil 1. Problem1'in doğruluk tablosu.

Mintermler cinsinden Y denklemi, $Y = \sum m(1,5,9,10,11,13)$ şeklinde ifade edilir. Mintermler üzerinden denklemin K-Map haritası Şekil 2'de gösterilmiştir.



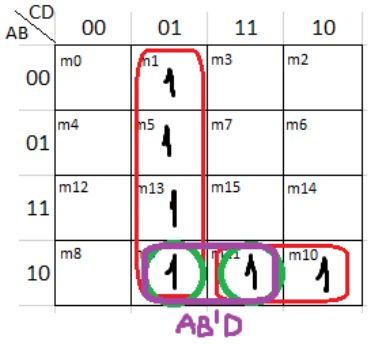
Şekil 2. Problem1 K-Map.

Glitch sistemdeki kısa ömürlü arızalardır. Bunları ‘hazard’ olarak da isimlendirebiliriz. K-map’te glitch tespit etmek için komşu (adjacent) mintermlere bakılır. Bunlar aynı grup içerisinde değilseler burada bir glitch vardır. Y denkleminin K-map’inden görüleceği üzere (Şekil 3) 1001 ve 1011 arasında glitch gözükmektedir.



Şekil 3. Problem1 K-Map Glitch

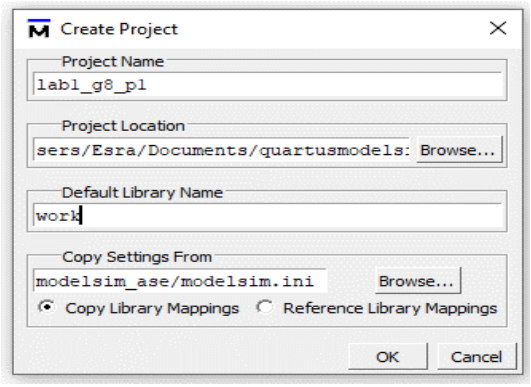
Bu glitchten kaynaklı sorunu çözmek için bu iki minterm arasında yeni bir grup oluşturabiliriz. Ondan gelecek olan ifadeyi de denklemimize ekleriz. Glitchsiz haliyle Y denklemi mintermler üzerinden Y = AB’C + C’D idi. Glitchi gidermek için oluşturduğumuz gruptan (Şekil 4) gelecek olan denklem AB’D ‘dir. Bunu da denklemimize eklersek son haliyle Y denklemi: Y = AB’C + C’D + AB’D olur.



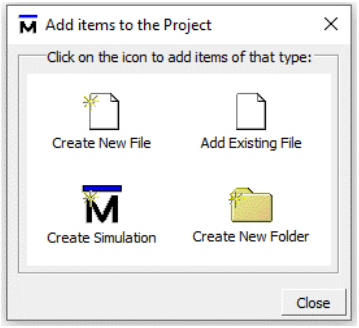
Şekil 4. Problem1 Glitch gruplaması.

2.1.2. Deneyin Yapılışı

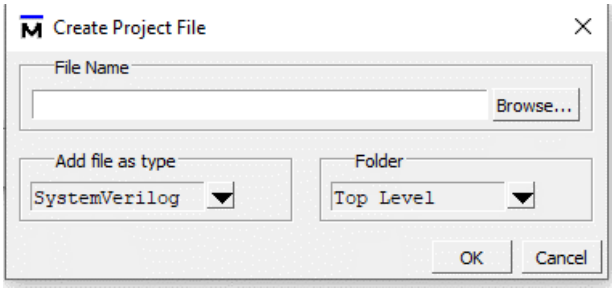
ModelSim’de File → Project kısmından yeni bir proje ekranı oluşturulur. Burada proje adı girilip dosyanın kaydedilmesi istenen yer belirlenir. “Create a New File” kısmından yeni dosya oluşturulur. Dosyanın tipi SystemVerilog olarak seçilir. Böylelikle kod dosyası oluşturulmuş olunur. Test Bench için de aynı adımlar izlenir ve onun için kod dosyası oluşturulur.



Şekil 5. ModelSim proje oluşturma ekranı.



Şekil 6. ModelSim dosya oluşturma ekranı.



Şekil 7. ModelSim dosya adı ve tipi belirleme ekranı.

Oluşturma tamamlandıktan sonra kodlar yazılır. Oluşturulan dosyalardan ilkinde giriş ve çıkış değişkenleri tanımlanır. Burada ayrıca boole cebir denklemi `assign` komutuyla tanımlanır. Tanımlarken '~' ifadesi logic NOT'ı, '&' ifadesi logic AND'i, '|' ifadesi logic OR'u simgelemektedir. Hazırlanan bu kod bloğuyla denklem ve değişkenler tanımlanmış olunur.

Test Bench koduyla denklemin logic çıktıları elde edilir. Bu kısımda giriş ihtimallerinin hepsi doğruluk tablosunda tanımlandığı gibi tanımlanır. Her giriş ihtimalinin sonuna 10 ns'lik beklemeler `#10` komutuyla konur. Blok yapısının sonuna `$stop;` konularak kodun durması sağlanır. Kodlar derlenir.

ModelSim denklem kodu:

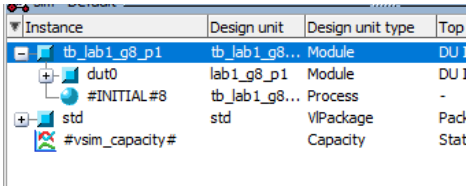
```
// Y = AB'C + C'D

`timescale 1ns/1ps

module lab2_g8_p1(

    input logic a,b,c,d,

    output logic y,y1,y2

);

assign #2 nb = ~b;

assign #2 nc = ~c;


assign #2 ac = a&c;

assign #2 anb_c = ac & nb;

assign #2 nc_d = nc & d;


assign #2 y = anb_c | nc_d; /*before fixing glitch  Y = AB'C + C'D*/

assign #2 ad = a & d;

assign #2 ad_nb = ad & nb;

assign #2 y1 = y | ad_nb; /*after fixing glitch seperate lag Y = AB'C
+ C'D +AB'D*/

assign #2 y2 =  anb_c| nc_d | ad_nb; /*after fixing glitch Y = AB'C +
C'D +AB'D*/

endmodule
```

ModelSim Test Bench kodu:

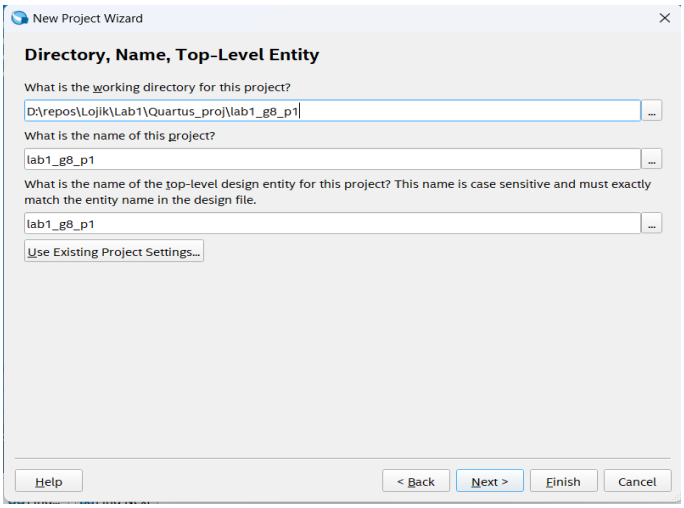
```
`timescale 1ns/1ps

module tb_lab2_g8_p1 ();

    logic a_tb, b_tb, c_tb, d_tb;

    logic y_tb,y1_tb,y2_tb;

    //logic y1_tb;


//lab2_g8_p1 dut0(a_tb, b_tb, c_tb, d_tb, y_tb);

lab2_g8_p1 dut0(a_tb, b_tb, c_tb, d_tb, y_tb, y1_tb, y2_tb);
```

```
initial begin
    a_tb = 0; b_tb = 0; c_tb = 0; d_tb = 0; #10
    a_tb = 0; b_tb = 0; c_tb = 0; d_tb = 1; #10
    a_tb = 0; b_tb = 0; c_tb = 1; d_tb = 0; #10
    a_tb = 0; b_tb = 0; c_tb = 1; d_tb = 1; #10
    a_tb = 0; b_tb = 1; c_tb = 0; d_tb = 0; #10
    a_tb = 0; b_tb = 1; c_tb = 0; d_tb = 1; #10
    a_tb = 0; b_tb = 1; c_tb = 1; d_tb = 0; #10
    a_tb = 0; b_tb = 1; c_tb = 1; d_tb = 1; #10
    a_tb = 1; b_tb = 0; c_tb = 0; d_tb = 0; #10
    a_tb = 1; b_tb = 0; c_tb = 0; d_tb = 1; #10
    a_tb = 1; b_tb = 0; c_tb = 1; d_tb = 0; #10
    a_tb = 1; b_tb = 0; c_tb = 1; d_tb = 1; #10
    a_tb = 1; b_tb = 1; c_tb = 0; d_tb = 0; #10
    a_tb = 1; b_tb = 1; c_tb = 0; d_tb = 1; #10
    a_tb = 1; b_tb = 1; c_tb = 1; d_tb = 0; #10
    a_tb = 1; b_tb = 1; c_tb = 1; d_tb = 1; #10
    $stop;
end
endmodule
```

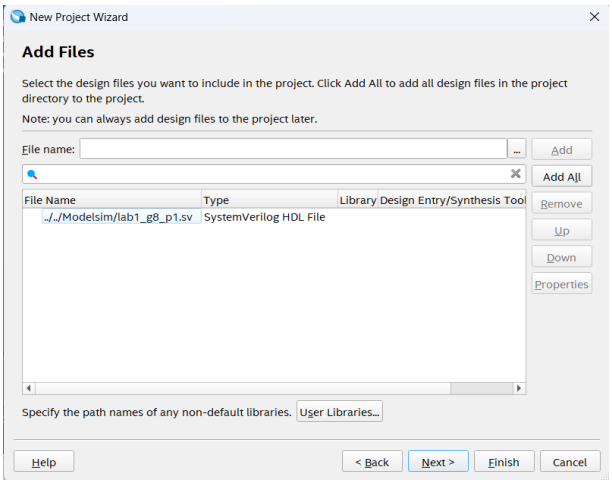
'work' adındaki library kısmında kod dosyaları gözükmektedir. Burada test bench dosyasına sağ tık yapılır ve simulate edilir. Şekil 8'de gözüken ekran açılır. Dosyaya sağ tık 'Add Wave' yapılarak test benchteki değişkenler dalga simülasyonuna eklenir. 'Run all' yapıldığında simülasyon hepsini koşturur ve bize lojik çıktıları verir.



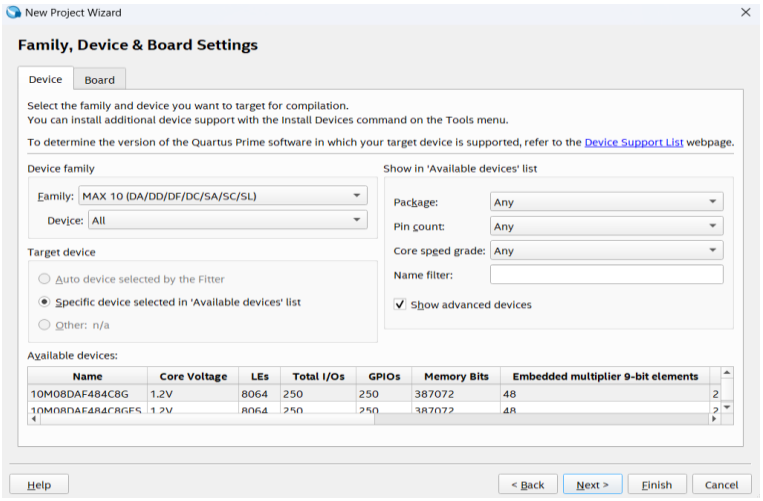
Şekil 8. ModelSim simülasyon test bench seçim ekranı.



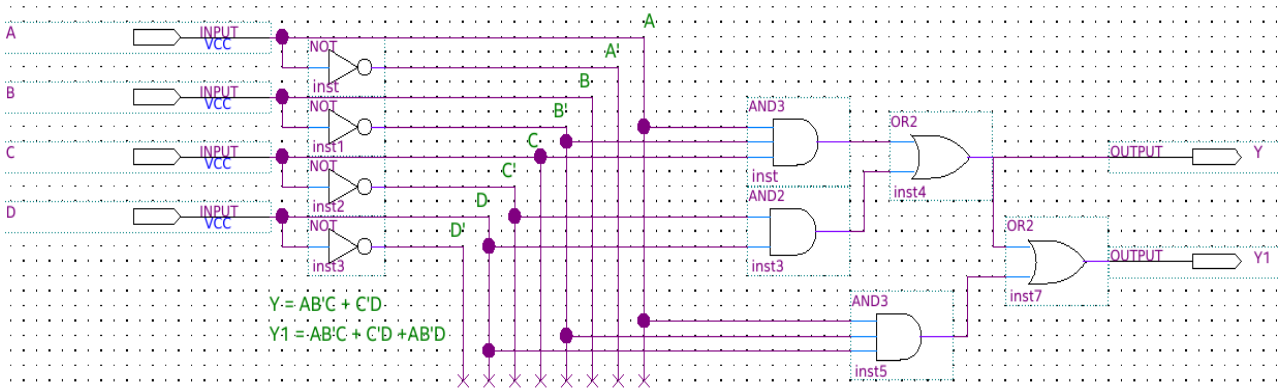
Şekil 9. Intel Quartus proje adı ve top-level design entity ve konumu belirleme ekranı.



Şekil 10. ModelSim'de oluşturulan systemverilog kodunun projeye eklenmesi.



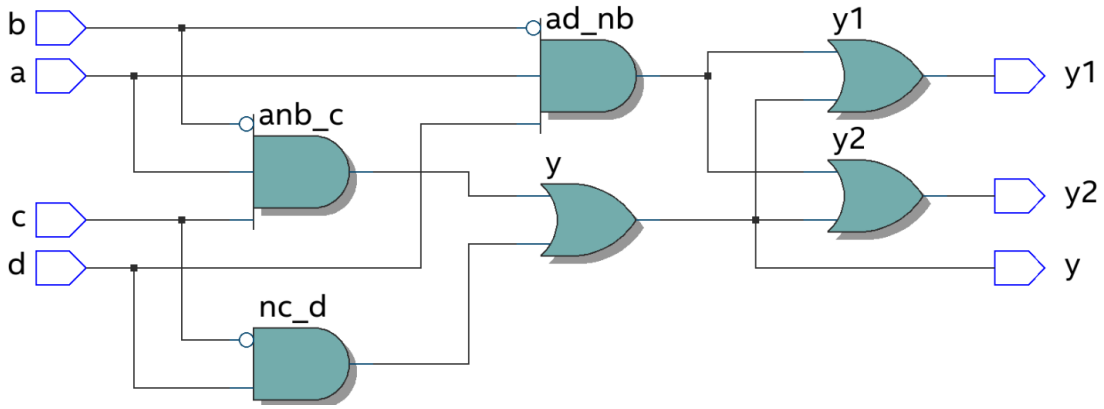
Şekil 11. Yeni proje oluştururken board seçimi.



Şekil 12. Pin atamaları yapılmış bir lojik devrenin şematik görüntüsü.

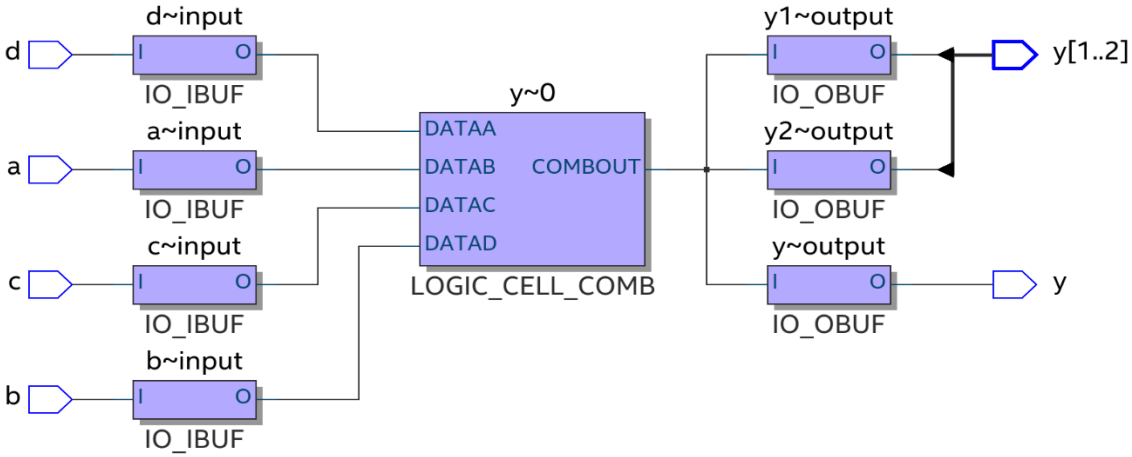
Şekil 9'da Intel Quartus üzerinden Problem 1 için lab2_g8_p1 adıyla proje oluşturulmuştur ve Şekil 10'da ModelSim'de oluşturulan projeden kod dosyası projeye eklenmiştir. Şekil 11'de ise cihaz ailesi seçimi MAX10 olarak yapılmıştır ve başka bir değişiklik yapılmadan devam edilerek proje oluşturulmuştur.

a) RTL Devre Şeması



Şekil 13. Problem1 Lojik devrenin RTL devre şeması.

b) Eşleştirme Sonrası Teknoloji Şeması



Şekil 14. Problem1 Lojik devrenin eşleştirme sonrası teknoloji şeması.

c) Analiz ve Sentez Özeti

Analysis & Synthesis Summary

<<Filter>>

Analysis & Synthesis Status	Successful - Thu May 4 21:44:06 2023
Quartus Prime Version	22.1std.1 Build 917 02/14/2023 SC Lite Edition
Revision Name	lab2_g8_p1
Top-level Entity Name	lab2_g8_p1
Family	MAX 10
Total logic elements	1
Total registers	0
Total pins	7
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Şekil 15. Analiz ve Sentez Özeti

Burdaki "Total logic elements" Intel Quartus'ta "Post Mapping" menüsünde veya üstte verilen eşleştirme sonrası teknoloji şemasında görülen "LOGIC_CELL_COMB" sayısını ifade etmektedir.

d) Analiz ve Sentez Kaynak Kullanım Özeti

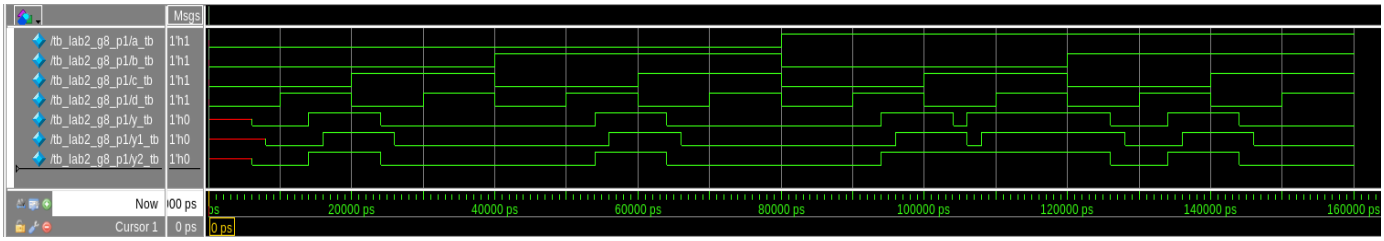
Analysis & Synthesis Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Estimated Total logic elements	1
2		
3	Total combinational functions	1
4	Logic element usage by number of LUT inputs	
1	-- 4 input functions	1
2	-- 3 input functions	0
3	-- <=2 input functions	0
5		
6	Logic elements by mode	
1	-- normal mode	1
2	-- arithmetic mode	0
7		
8	Total registers	0
1	-- Dedicated logic registers	0
2	-- I/O registers	0
9		
10	I/O pins	7
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	y~0
15	Maximum fan-out	3
16	Total fan-out	14
17	Average fan-out	0.93

Şekil 16. Analiz ve Sentez Kaynak Kullanım Özeti

e) Simülasyon Dalga Sonuçları



Şekil 17. Simülasyon dalga sonuç ekranı.

Simülasyonda başta çıkan kırmızı çizgiler değeri bilinmeyen bir x'i temsil etmektedir. Kodda simüle edilen propagasyon gecikmesi nedeniyle çıktılar girişlere kıyasla gecikmeli gelmekte o nedenle gecikme süresince önceki çıktıyı göstermektedir. Ancak ilk başta çıktı olmadığından bilinmeyen bir x simülasyonda çıktı olarak gelmiştir. 1001 ve 1011 arasında glitch gözükmektedir. Bunu dalgadaki anlık 101 geçişi yaşanmasından anlamaktayız. Sonradan eklenen OR gate ayrı bir lag verildiğinde glitch düzelmemiştir. Bunun için y2 çıktısında iki OR kapısına da aynı anda lag verilmiştir. Bu sayede glitchin düzeldiği gözlemlenmiştir.

2.1.3. Sonuçların Yorumu

Devre başlatıldığında input gecikmesi olduğundan çıkış değerleri bu gecikme sırasında bilinmeyen 'x' olarak gözükmektedir. K-Map'lerden de görüldüğü üzere glitche neden olan komşu mintermler gruplandığında glitchin düzeldiği gözlemlenmiştir.

2.2. Problem II

2.2.1. Teorik Araştırma

Decoder (kod çözücü) n tane girişe karşılık 2^n çıktı sağlayan devrelerdir. Bu çıktılar mintermlere karşılık gelebilmektedir. Kodlanmış bilgileri anlaşılır hale dönüştürmek için kullanılır.

Don't care (önemseme) koşulu, bir değişkenler grubu oluşturmak için bir K-haritasının boş hücrelerini kullanabileceğimizi söyler. [2] Bunların 0 veya 1 olması çıkışı etkilemez. Bunları K-Map'te işimize yarayacak şekilde 0 veya 1 olarak kabul ederiz. Denklemi daha sade yapabilmek için K-Map gruplarını en geniş halinde tutmamız gerekmektedir. Don't care koşullarını bu genişliği sağlayabilmek için 1 olarak alırız.

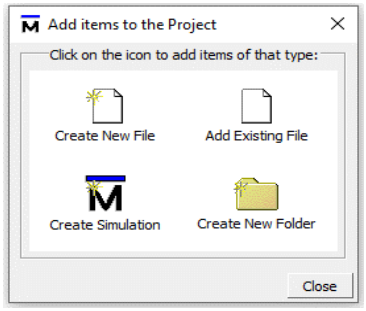
Seven Segment LED Display'in doğruluk tablosu Şekil 18'de verilmiştir.

x3	x2	x1	x0	A	B	C	D	E	F	G	
0	0	0	0	0	0	0	0	0	0	1	m0
0	0	0	1	X	X	X	X	X	X	X	m1
0	0	1	0	1	0	0	1	1	1	1	m2
0	0	1	1	1	1	1	1	0	0	1	m3
0	1	0	0	X	X	X	X	X	X	X	m4
0	1	0	1	0	0	0	1	1	1	0	m5
0	1	1	0	X	X	X	X	X	X	X	m6
0	1	1	1	X	X	X	X	X	X	X	m7
1	0	0	0	X	X	X	X	X	X	X	m8
1	0	0	1	X	X	X	X	X	X	X	m9
1	0	1	0	X	X	X	X	X	X	X	m10
1	0	1	1	X	X	X	X	X	X	X	m11
1	1	0	0	1	0	0	1	1	1	0	m12
1	1	0	1	1	1	0	1	1	0	1	m13
1	1	1	0	1	0	1	1	0	1	1	m14
1	1	1	1	X	X	X	X	X	X	X	m15

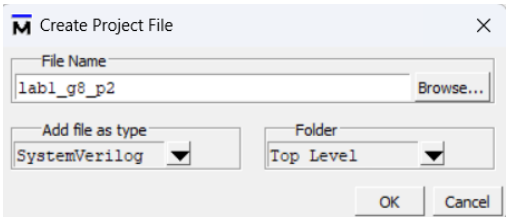
Şekil 18. Problem2 doğruluk tablosu.

2.2.2. Deneyin Yapılışı

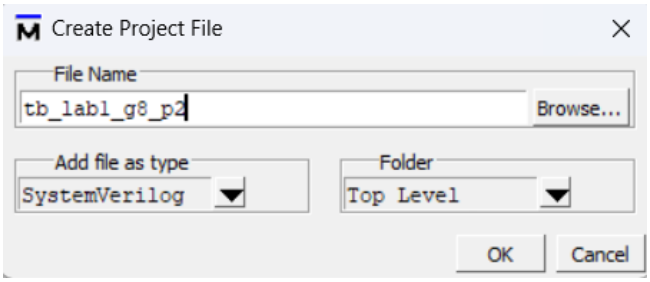
ModelSim'de File → Project kısmından yeni bir proje ekranı oluşturulur. Burada proje adı girilip dosyanın kaydedilmesi istenen yer belirlenir. "Create a New File" kısmından yeni dosya oluşturulur. Dosyanın tipi SystemVerilog olarak seçilir. Böylelikle kod dosyası oluşturulmuş olunur. Test Bench için de aynı adımlar izlenir ve onun için kod dosyası oluşturulur.



Şekil 19. ModelSim dosya oluşturma ekranı.



Şekil 20. ModelSim dosya adı ve tipi belirleme ekranı.

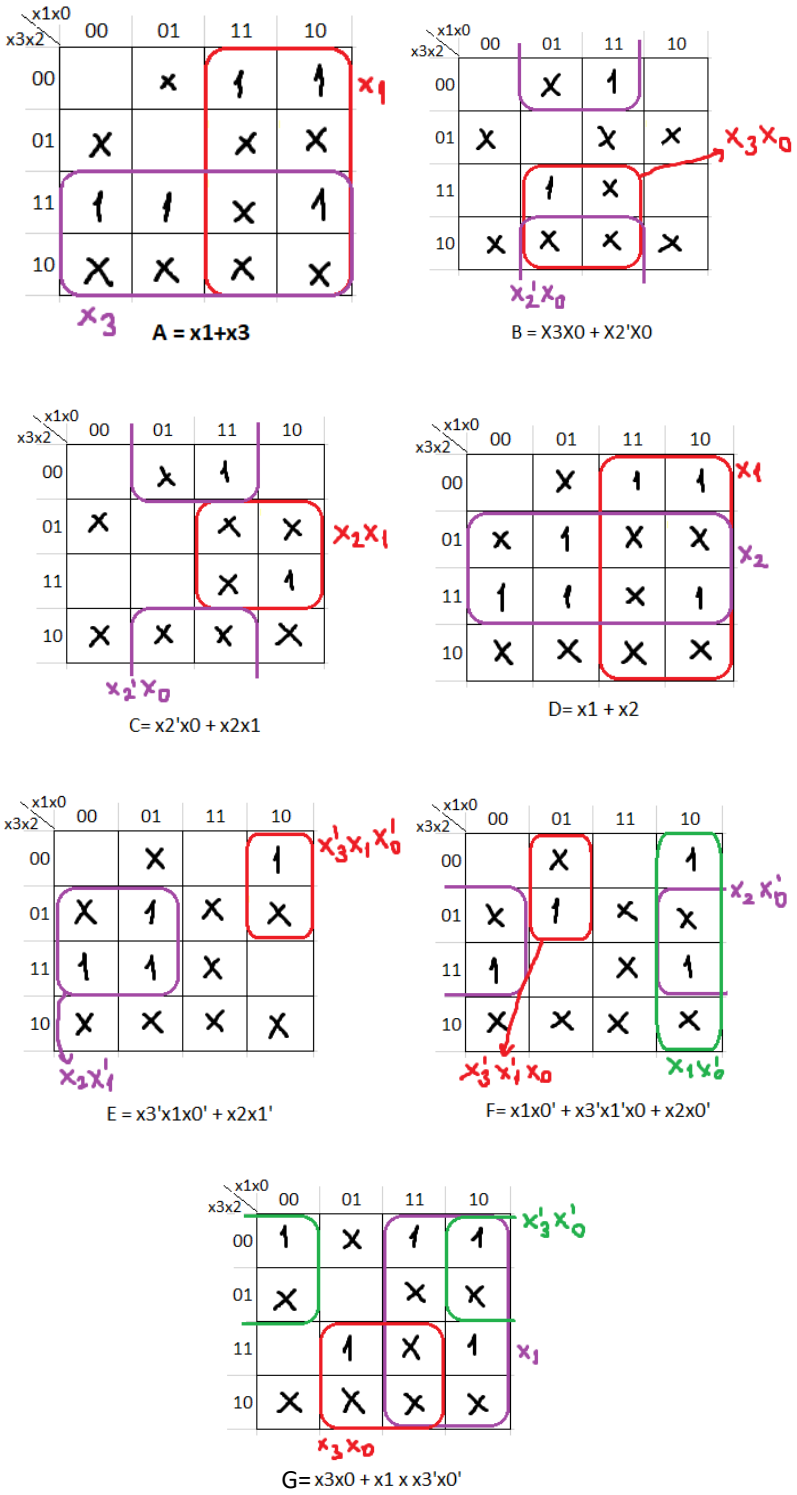


Şekil 21. ModelSim Test Bench dosya adı ve tipi belirleme ekranı.

Oluşturma tamamlandıktan sonra kodlar yazılır. Oluşturulan dosyalardan ilkinde giriş ve çıkış değişkenleri tanımlanır. Burada ayrıca boole cebir denklemi `assign` komutuyla tanımlanır. Tanımlarken '~' ifadesi logic NOT'ı, '&' ifadesi logic AND'i, '|' ifadesi logic OR'u simgelemektedir. Hazırlanan bu kod bloğuyla denklem ve değişkenler tanımlanmış olunur.

Test Bench koduyla denklemin logic çıktıları elde edilir. Bu kısımda giriş ihtimallerinin hepsi doğruluk tablosunda tanımlandığı gibi tanımlanır. Her giriş ihtimalinin sonuna 10 ns'lik beklemeler `#10` komutuyla konur. Blok yapısının sonuna `$stop;` konularak kodun durması sağlanır. Kodlar derlenir.

Tüm çıktılar K-Map haritalarıyla sadeleştirilmiştir. Çıktıların haritaları ve sadeleşmiş denklemleri aşağıda verilmiştir.



Şekil 22. Problem2 Çıktıların K-Map haritaları.

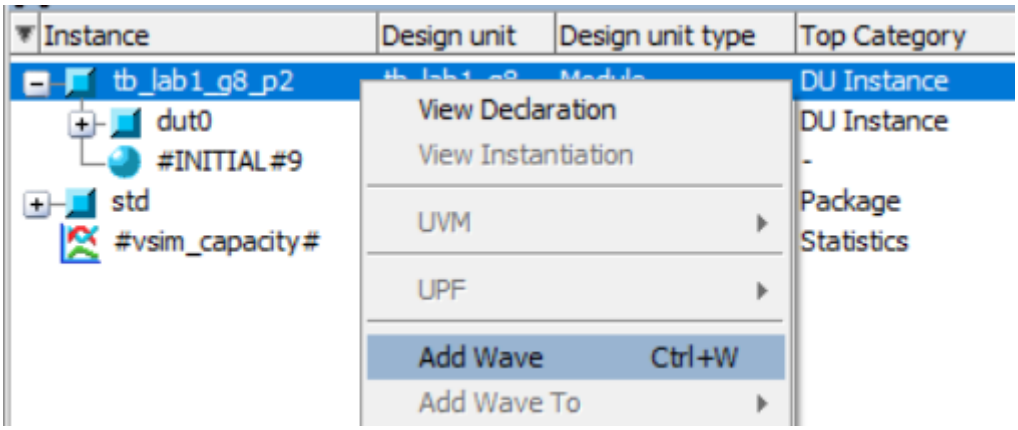
ModelSim denklem kodu:

```
/*
A = x1+x3
B= x3x0+x2'x0
C=x2'x0+x2x1
D=x1+x2
E=x3'x1x0'+x2x1'
F=x1x0'+x3'x1'x0+x2x0'
G=x0x3+x1+x0'x3'
*/
`timescale 1ns/1ps
module lab2_g8_p2(
    input logic x0,x1,x2,x3,
    output logic A,B,C,D,E,F,G
);
assign A = x1 | x3;
assign B = x0&x3 | x0&~x2;
assign C = x0&~x2 | x1&x2;
assign D = x1 | x2;
assign E = ~x0&x1&~x3 | ~x1&x2;
assign F = ~x0&x1 | x0&~x1&~x3 | ~x0&x2;
assign G = x0&x3 | x1 | ~x0&~x3;
endmodule
```

ModelSim Test Bench kodu:

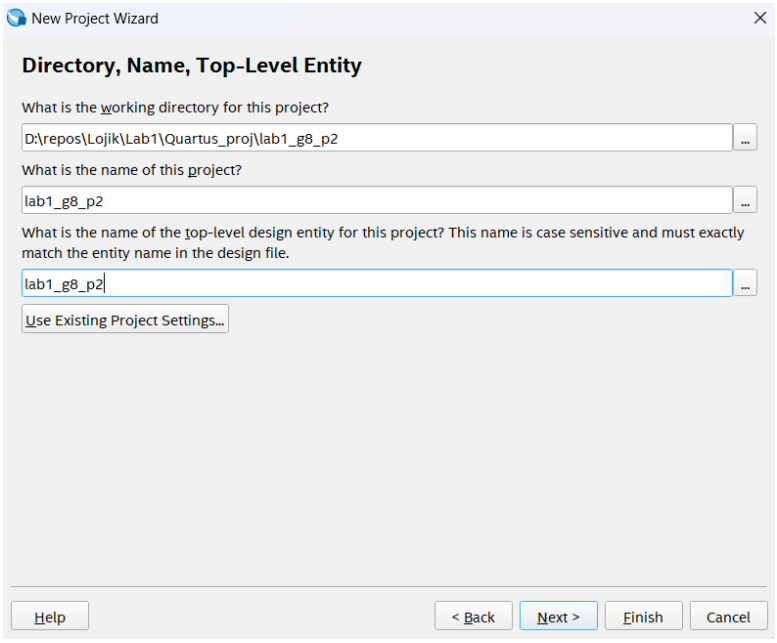
```
      `timescale 1ns/1ps

      module tb_lab2_g8_p2 ();

          logic x0_tb, x1_tb, x2_tb, x3_tb;

          logic A_tb, B_tb, C_tb, D_tb, E_tb, F_tb, G_tb;

      lab2_g8_p2 dut0(x0_tb, x1_tb, x2_tb, x3_tb, A_tb, B_tb, C_tb,
D_tb, E_tb, F_tb, G_tb);

      initial begin

          x3_tb = 0; x2_tb = 0; x1_tb = 0; x0_tb = 0; #10

          x3_tb = 0; x2_tb = 0; x1_tb = 0; x0_tb = 1; #10

          x3_tb = 0; x2_tb = 0; x1_tb = 1; x0_tb = 0; #10

          x3_tb = 0; x2_tb = 0; x1_tb = 1; x0_tb = 1; #10

          x3_tb = 0; x2_tb = 1; x1_tb = 0; x0_tb = 0; #10

          x3_tb = 0; x2_tb = 1; x1_tb = 0; x0_tb = 1; #10

          x3_tb = 0; x2_tb = 1; x1_tb = 1; x0_tb = 0; #10

          x3_tb = 0; x2_tb = 1; x1_tb = 1; x0_tb = 1; #10

          x3_tb = 1; x2_tb = 0; x1_tb = 0; x0_tb = 0; #10

          x3_tb = 1; x2_tb = 0; x1_tb = 0; x0_tb = 1; #10

          x3_tb = 1; x2_tb = 0; x1_tb = 1; x0_tb = 0; #10

          x3_tb = 1; x2_tb = 0; x1_tb = 1; x0_tb = 1; #10

          x3_tb = 1; x2_tb = 1; x1_tb = 0; x0_tb = 0; #10

          x3_tb = 1; x2_tb = 1; x1_tb = 0; x0_tb = 1; #10

          x3_tb = 1; x2_tb = 1; x1_tb = 1; x0_tb = 0; #10

          x3_tb = 1; x2_tb = 1; x1_tb = 1; x0_tb = 1; #10

          $stop;

      end

      endmodule
```

‘work’ adındaki library kısmında kod dosyaları gözükmektedir. Burada test bench dosyasına sağ tık yapılır ve simulate edilir. Şekil 23’te gözüken ekran açılır. Dosyaya sağ tık ‘Add Wave’ yapılarak test benchteki değişkenler dalga simülasyonuna eklenir. ‘Run all’ yapıldığında simülasyon hepsini koşturur ve bize lojik çıktıları verir.

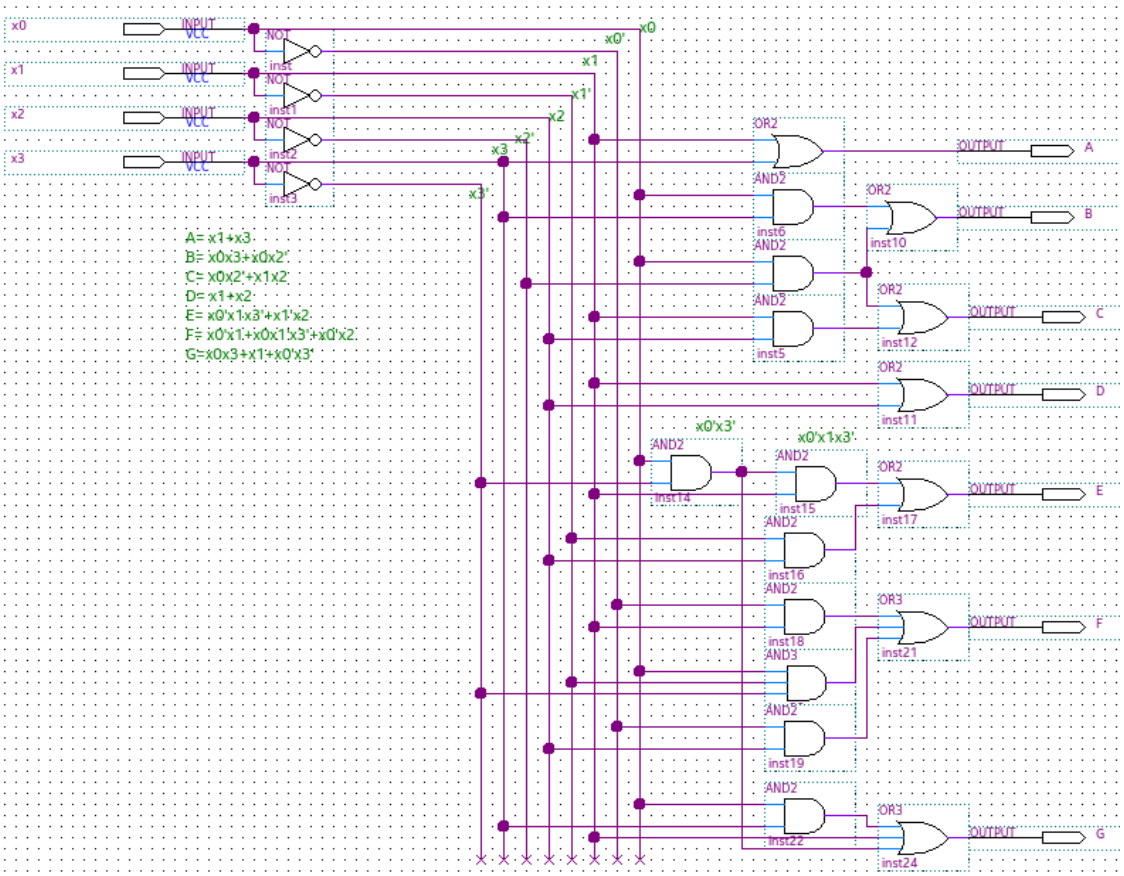


Şekil 23. ModelSim simülasyon test bench seçim ekranı.

Intel Quartus üzerinden oluşturulan projeye "lab2_g8_p2" adı verilip problem 1'deki aynı prosedür izlenerek ModelSim'de oluşturulan proje dosyası Quartus'ta oluşturulan projeye dahil edilmiştir. Ardından problem 1'de olduğu gibi MAX10 cihaz ailesi seçilerek proje oluşturulmuştur.



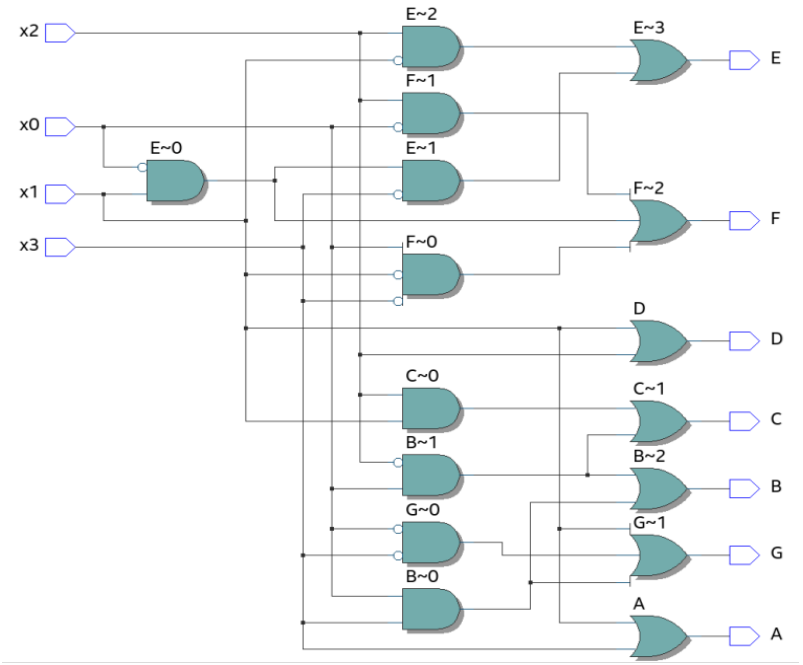
Şekil 24. Intel Quartus proje adı, top-level design entity ve konumu belirleme ekranı.



Şekil 25. Pin atamaları yapılmış bir lojik devrenin şematik görüntüsü.

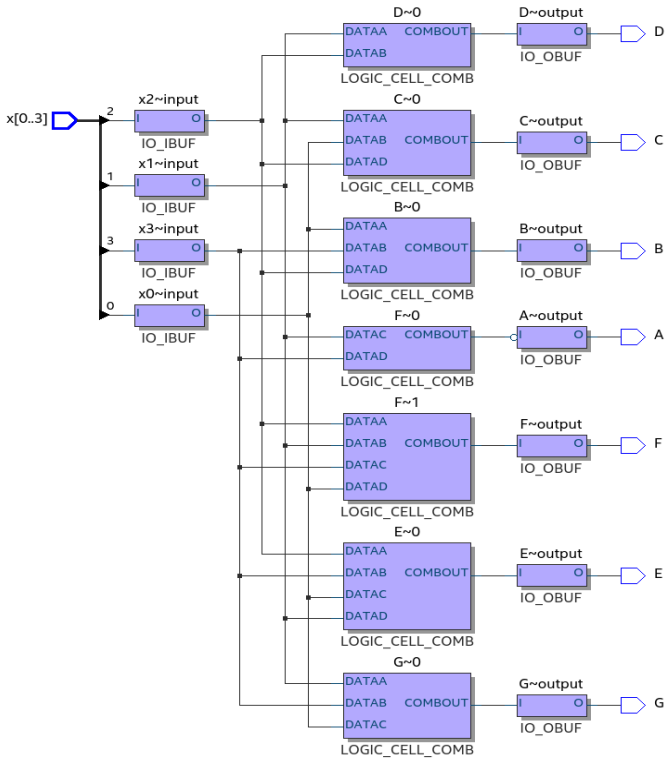
Devrede 4 NOT kapısı, 10 AND kapısı ve 7 OR kapısı olmak üzere toplamda 21 tane devre elemanı kullanılmıştır.

a) RTL Devre Şeması



Şekil 26. Lojik devrenin RTL devre şeması.

b) Eşleştirme Sonrası Teknoloji Şeması



Şekil 27. Lojik devrenin eşleştirme sonrası teknoloji şeması

11 tane input-output buffer kullanılmıştır. Bunlar Şekil 28'deki Total pins'e karşılık gelmektedir. 7 tane de lojik eleman kullanılmıştır. Şekil 28'deki sentez özetinde Total logic elements olarak görülmektedir.

c) Analiz ve Sentez Özeti

Analysis & Synthesis Summary

<<Filter>>

Analysis & Synthesis Status	Successful - Thu May 4 23:14:51 2023
Quartus Prime Version	22.1std.1 Build 917 02/14/2023 SC Lite Edition
Revision Name	lab2_g8_p2
Top-level Entity Name	lab2_g8_p2
Family	MAX 10
Total logic elements	7
Total registers	0
Total pins	11
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Şekil 28. Analiz ve Sentez Özeti Problem2.

Buradaki “Total logic elements” Intel Quartus’ta “Post Mapping” menüsünde veya üstte verilen eşleştirme sonrası teknoloji şemasında görülen “LOGIC_CELL_COMB” sayısını ifade etmektedir.

d) Analiz ve Sentez Kaynak Kullanım Özeti

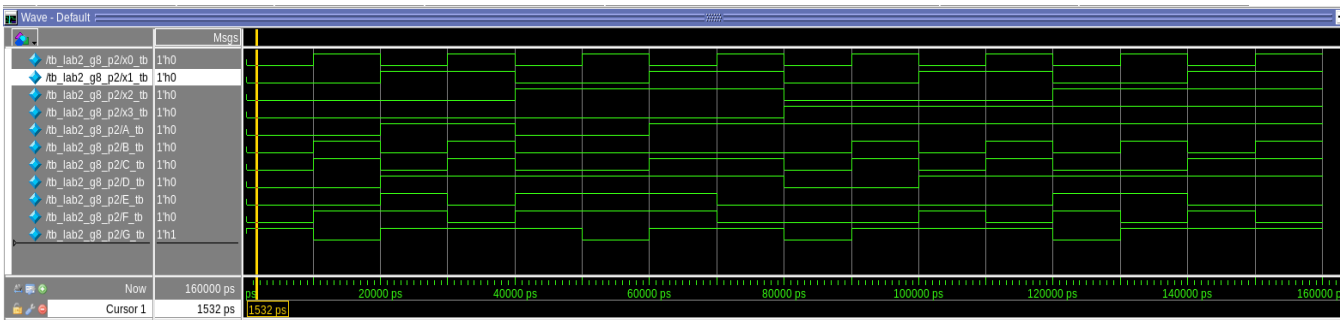
Analysis & Synthesis Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Estimated Total logic elements	7
2		
3	Total combinational functions	7
4	Logic element usage by number of LUT inputs	
1	-- 4 input functions	2
2	-- 3 input functions	3
3	-- <=2 input functions	2
5		
6	Logic elements by mode	
1	-- normal mode	7
2	-- arithmetic mode	0
7		
8	Total registers	0
1	-- Dedicated logic registers	0
2	-- I/O registers	0
9		
10	I/O pins	11
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	x1~input
15	Maximum fan-out	6
16	Total fan-out	39
17	Average fan-out	1.34

Şekil 29. Analiz ve Sentez Kaynak Kullanım Özeti Problem2.

e) Simülasyon



Şekil 30. Simülasyon dalga sonuç ekranı.

2.2.3. Sonuçların Yorumu

Simülasyon sonuçlarında eşleştirme sonrası teknoloji şemasında görmekteyiz ki 11 tane input-output buffer kullanılmıştır. Bunlar Şekil 28'deki Total pins'e karşılık gelmektedir. 7 tane de lojik eleman kullanılmıştır. Şekil 28'deki sentez özetinde Total logic elements olarak görülmektedir. RTL şemasında 9 AND kapısı 7 tane de OR kapısı görülmektedir.

Problem 1'e kıyasla burada daha fazla output yer almaktadır. Bu nedenle de daha fazla kapı ve lojik eleman kullanıldığından daha fazla yer kaplamaktadır. Devrenin dalga simülasyonunda problem 1'in aksine glitch oluşumu gözlenmemiştir.

3. Sonuçlar ve Genel Yorumlar

Propagasyon gecikmesi olduğunda girişte tanımlanamayan bir x oluşumu gözlenmiştir. Glitche rastlandığında karnough mapte komşu olan mintermlerin gruplanması da gerektiği farkedilmiştir. Bu sayede oluşabilecek anlık hata sıkıntılarının da önüne geçilmiştir.

Seven segment LCD Display'in lojik kapılarla gerçeklemesi yapılmıştır.

4. Referanslar

[1] URL: https://haktanbozer.com.tr/nedir/yayin-gecikmesi-nedir/ Accessed: 04.05.2023

[2] URL: https://www.javatpoint.com/dont-care-condition-in-k-map-in-digital-electronics Accessed: 04.05.2023