



GEBZE TEKNİK ÜNİVERSİTESİ
ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 3 Deney Raporu

Donanım Tanımlama Dilleri Uygulamaları: Birleşik Devreler

Hazırlayanlar
1) 200102002025 – Umut Mehmet ERDEM
2) 200102002066 – Emre TANER

İçindekiler

1. Giriş	2
2. Problemler	2
3. Sonuçlar ve Genel Yorumlar	23
4. Referanslar	23

1. Giriş

Bu deneyde aşağıda verilen maddeler amaçlanmaktadır:

1. Donanım tanıma dillerini (DTD) kullanarak devre tasarımı yapmak.
2. Sentezleyici araçları kullanarak, DTD ile tanımlanan devreleri FPGA için sentezlemek.
3. Simülasyon araçları kullanarak, otomatik simülasyon yaptırmak.
4. Devrede oluşan gecikmeleri gözlemlemek.

2. Problemler

2.1. Problem I – $2^N \times 1$ Çoğullayıcı (MUX) Tasarımı

2.1.1. Teorik Araştırma

MUX'un tanımı şu şekildedir:

Elektronikte çoklayıcı, birden fazla analog veya sayısal veri kaynağından birini seçerek o kaynağı çıktı olarak tek bir kanala ileten sistemlerdir.

Bir çoklayıcı birden fazla girişten tek bir çıkışı sağlayan bir anahtar işlevi görür. Çoklayıcıya bağlanan seçici, girdilerden yalnızca bir tanesini seçerek çıkışa yönlendirir. Çoklayıcı ikizkenar yamuk olarak gösterilir; uzun olan paralel kenarı giriş iğnelerini, kısa olan paralel kenarı ise çıkış iğnesini bulundurmaktadır.

2.1.2. Deneyin Yapılışı

Quartus Prime üzerinden yeni bir proje açılmış açılan projenin üst düzey tasarım varlığının adı (name of top-level design entity) Modelsim programı üzerinden oluşturulan System Verilog (.sv) dosyası ile aynı şekilde isimlendirilmiştir. İleriki adımlarda “.sv” dosyası ve kullanılacak board Şekil 1’ deki gibi seçilmiştir.

New Project Wizard

Family, Device & Board Settings

Device Board

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: MAX 10 (DA/DF/DC/SA/SC)

Device: All

Target device

☐ Auto device selected by the Fitter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter:

☒ Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit elements
10M08DAF484C8G	1.2V	8064	250	250	387072	48

< >

< Back Next > Finish Cancel Help

Şekil 1.Yeni Proje Oluştururken Board Seçimi

a) $2^N \times 1$ MUX Devresini Donanım Tanımlama Dili Kullanarak Gerçeklenmesi

$2^N \times 1$ MUX devresini DTD kullanarak kapı düzeyinde sadece NAND kapıları kullanarak tasarlandı. Yazılan kod aşağıdaki gibidir:

```
/* lab3_g7_p1.sv
* Hazırlayanlar: Umut Mehmet ERDEM – Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 – Problem 1
*  $2^N \times 1$  MUX Devresini DTD Dili Kullanarak Gerçeklenmesi
*/
`timescale 1ns/1ps
module lab3_g7_p1(
    input logic A,
    input logic [1:0] B,
    output logic C
);
    mux2 mux_1(A, B[1:0], C);
endmodule

module mux2(
    input logic X,
    input logic [1:0] Y,
    output logic Q
);
    assign Q = ~(~(X&Y[1])&(~(~(X&X)&Y[0])));
endmodule
```

Yazılan kodda, ana(lab3_g7_p1) modül içerisinde kullanılacak “mux2” modülü tanımlanmıştır. “mux2” modülünde; X seçim bitini, iki bitlik Y değişkeni giriş bitlerini ve Q ise çıkışı temsil etmektedir. Seçim bitinin görevi hangi girişin çalışacağını göstermektedir. “assign” ile Q çıkışına K-map üzerinden bulunan denklem atama yapılmıştır. 2×1 MUX doğruluk tablosu Tablo 1’deki gibi yapılmıştır. Ana modülde; A, B, C parametreleri tanımlanmış, tanımlanan parametreler “mux2” modülü olarak oluşturulan “mux_1” e verilmiştir.

X	Y1	Y0	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Tablo 1. 2×1 Çoğullayıcının Doğruluk Tablosu

Elde edilen doğruluk tablosundaki Q çıkışlarını Karnaugh Map yardımıyla sadeleştirilmiş ve Q çıkışı elde edilmiştir.

X/Y ₁ Y ₀	Y ₁ 'Y ₀ '	Y ₁ 'Y ₀	Y ₁ Y ₀	Y ₁ Y ₀ '
X'	0	1	1	0
X	0	0	1	1

Tablo 2. Q Çıkışlarını K-Map Yardımıyla Sadeleştirilmesi

K-Map sonucunda $Q = XY_1 + X'Y_0$ denklemi elde edilmiştir. Problem 1'de NAND kapıları kullanarak MUX oluşturulması istenildiğinden, elde edilen boole denklemi NAND kapıları kullanılacak şekilde $((XY_1)'((XX)'Y_0)')$ denklemine dönüştürülmüştür ve kodda Q çıkışına yazılmıştır.

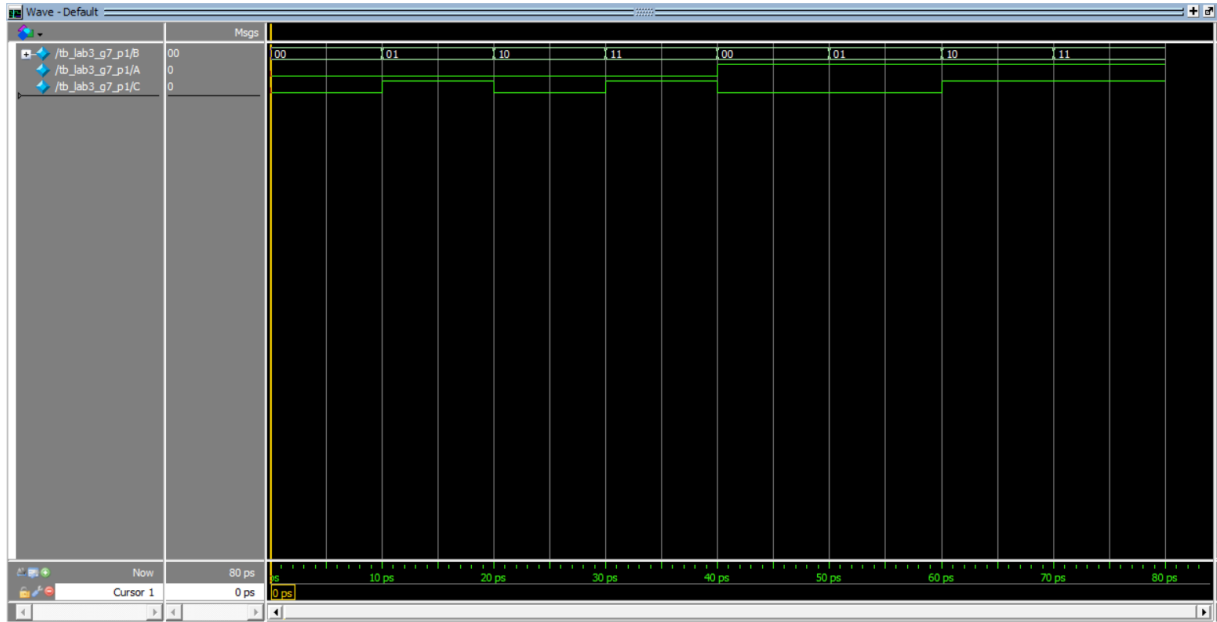
b) Tasarlanılan DTD Modelinin Simüle Edilmesi ve Zamanlama Diyagramı

Tasarlanılan DTD modelini ve ilgili fonksiyonel test tezgâhı kodunu girişlere bütün olası kombinasyonları her 10ns'lik adımlarda uygulanmıştır ve çıkış sinyali gözlemlenecek şekilde simüle edilmiştir. Yazılan kod aşağıdaki gibidir:

```

/* tb_lab3_g7_p1.sv
* Hazırlayanlar: Umut Mehmet ERDEM - Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 - Problem 1 Testbench
*  $2^N \times 1$  MUX Devresini DTD Dili Kullanarak Gerçeklenmesi
* Bütün olası girişlere göre çıkış gözlemlenir.
*/
// Zaman birimi ve simülasyon çözünürlüğü
`timescale 1ns/1ps
module tb_lab3_g7_p1();
// Test tezgahlarında port bulunmaz
logic [1:0]B; // test tezgahi giriş sinyal tanımları
logic A, C; // test tezgahi çıkış sinyal tanımları
// Test edilecek modülün yaratımı ve port bağlantılarının yapılması
// dut = device under test
    lab3_g7_p1 dut0(A, B, C);
// Bu kısımda sinyaller test edilen devreye sıralı olarak uygulanır.
// Sonuçlar test edilen devre çıkışlarında gözlemlenebilir.
initial begin
    A = 0; B[1] = 0; B[0] = 0;      #10 // 000 -10 ns bekle
    B[0]=1;                          #10 //001
    B[1]=1; B[0]=0;                  #10 //010
    B[0]=1;                          #10 //011
    A=1; B[1]=0; B[0]=0;             #10 //100
    B[0]=1;                          #10 //101
    B[1]=1; B[0]=0;                  #10 //110
    B[0]=1;                          #10 //111
    $stop; // simülasyonu durdur
end
endmodule

```



Şekil 2. 2×1 Çoğullayıcının Zamanlama Diyagramı

2.1.3. Sonuçların Yorum

2×1 MUX oluştururken 2 giriş, bir seçici ve bir çıkış olarak tasarlandı. Yapılan araştırmalara göre giriş sayısı 2'nin katları şeklinde artarken seçicilerde girişlerdeki her 2 kat artışta bir arttığı gözlemlendi. Problem 1'de istenilen MUX başarılı bir şekilde tasarlandı ve rapora eklendi.

2.2. Problem II- $2^N \times 1$ Çoğullayıcı (MUX) ile Temel Lojik Kapıların ve Büyük Çoğullayıcıların Tasarımı

2.2.1. Deneyin Yapılışı

Quartus Prime üzerinden yeni bir proje açılmış açılan projenin üst düzey tasarım varlığının adı (name of top-level design entity) Modelsim programı üzerinden oluşturulan System Verilog (.sv) dosyası ile aynı şekilde isimlendirilmiştir. İleriki adımlarda ".sv" dosyası ve kullanılacak board Şekil 3' deki gibi seçilmiştir.

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit elements
10M08DAF484C8G	1.2V	8064	250	250	387072	48

Şekil 3. Yeni Proje Oluştururken Board Seçimi

a) $2^N \times 1$ MUX ile Donanım Tanımlama Dili Kullanarak AND, OR, NAND, NOR Yapımı

Problem 1 'deki $2^N \times 1$ MUX'ı kullanarak AND, OR, NAND, NOR kapıları için tasarlanmıştır. Yazılan kod aşağıdaki gibidir:

```
/* lab3_g7_p2.sv
* Hazırlayanlar: Umut Mehmet ERDEM - Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 - Problem 2
*  $2^N \times 1$  Çoğullayıcı ile Temel Lojik Kapıların Tasarımı
*/
`timescale 1ns/1ps
module lab3_g7_p2(
    input logic X,
    input logic [1:0] Y_and, Y_or, Y_nand, Y_nor,
    output logic Q_and, Q_or, Q_nand, Q_nor
);
endmodule
```

Ana modülde X seçme biti; Y_and, Y_or, Y_nand, Y_nor iki bitlik giriş bitleri; Q_and, Q_or, Q_nand, Q_nor çıkış bitlerini temsil etmektedir. AND, OR, NAND, NOR için kullanılacak "mux2" modülleri testbench dosyasında tanımlanmış, tanımlanan X ve Y portlarına atamalar yapılarak Q çıkışları bulunmuştur.

b) Temel Kapılar İçin Tasarlanılan DTD Modelinin Simüle Edilmesi

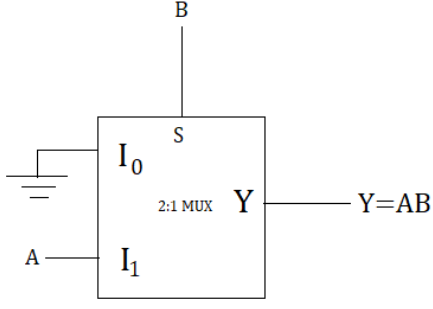
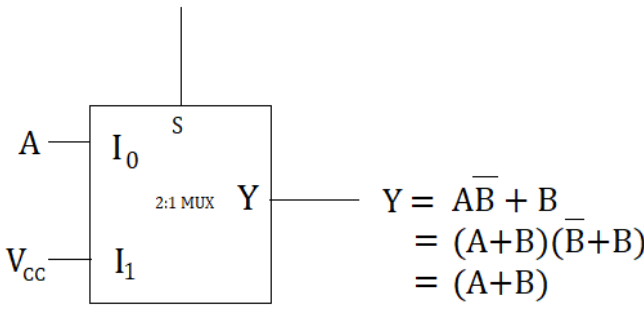
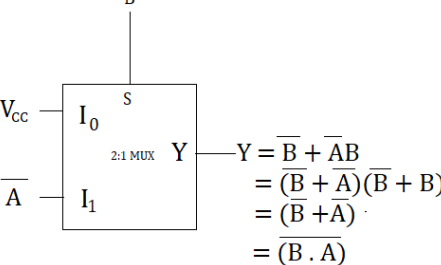
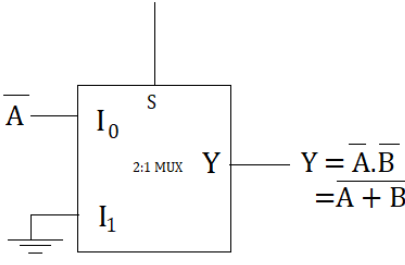
```
/* tb_lab2_g7_p2.sv
* Hazırlayanlar: Umut Mehmet ERDEM - Emre TANER
* Notlar: ELM235 2023 Bahar Lab2 - Problem 2 Testbench
*  $2^N \times 1$  Çoğullayıcı ile Temel Lojik Kapıların Tasarımı
* Bütün olası girişlere göre çıkış gözlemlenir.
*/
// Zaman birimi ve simülasyon çözünürlüğü
`timescale 1ns/1ps
module tb_lab3_g7_p2();
// Test tezgahlarında port bulunmaz
    logic X;
    logic [1:0] Y_and, Y_or, Y_nand, Y_nor;
    logic Q_and, Q_or, Q_nand, Q_nor;
    mux2 mux_and(X, Y_and[1:0], Q_and);
    mux2 mux_or(X, Y_or[1:0], Q_or);
    mux2 mux_nand(X, Y_nand[1:0], Q_nand);
    mux2 mux_nor(X, Y_nor[1:0], Q_nor);
// Test edilecek modülün yaratımı ve port bağlantılarının yapılması
// dut = device under test
    lab3_g7_p2 dut0(X, Y_and, Y_or, Y_nand, Y_nor, Q_and, Q_or, Q_nand, Q_nor);
// Bu kısımda sinyaller test edilen devreye sıralı olarak uygulanır.
// Sonuçlar test edilen devre çıkışlarında gözlenebilir.
```

Yukarıda belirttiğimiz gibi temel kapılar için olan "mux2" modülleri; "mux_and", "mux_or", "mux_nand", "mux_nor" olarak tanımlanmış ve parametreleri girilmiştir. Simüle edilebilmesi için dut0 belirtilmiş ve zamanlama diyagramı parametreleri verilmiştir.

```

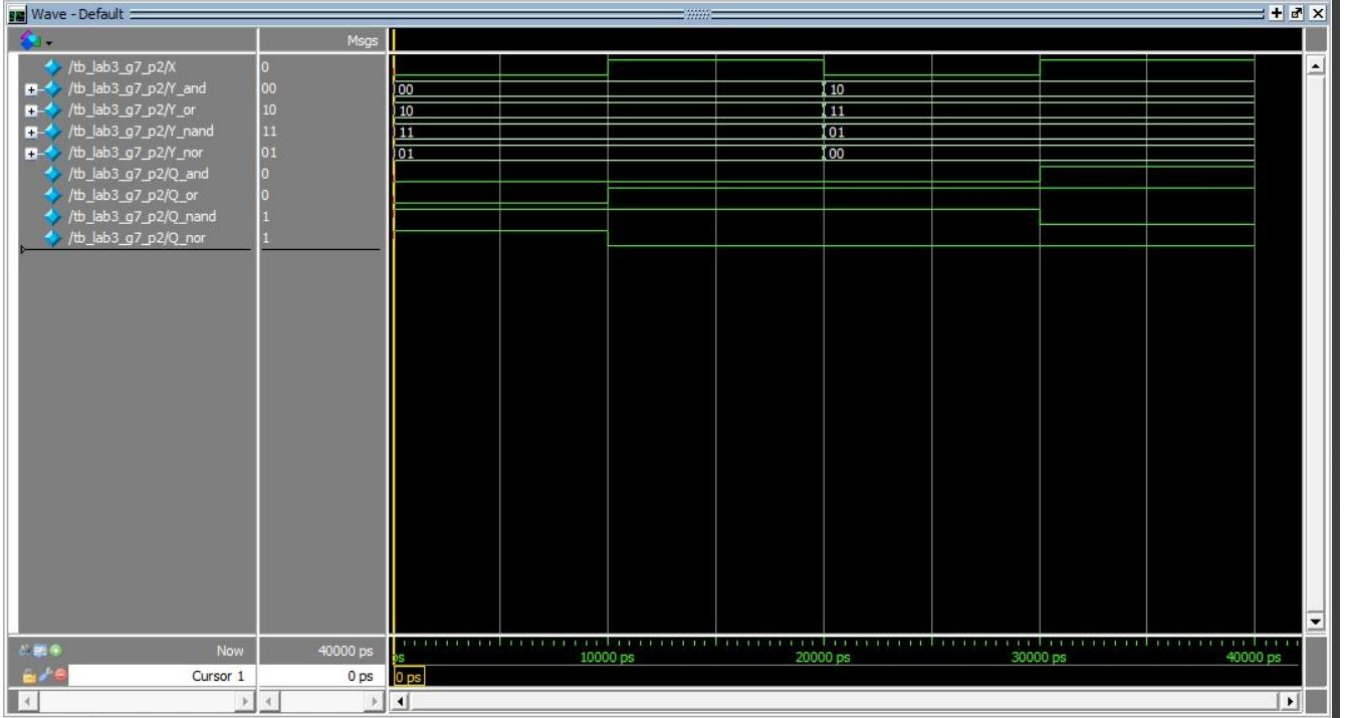
initial begin
    X=0; Y_and[0]=0; Y_and[1]=0; Y_or[0]=0; Y_or[1]=1; Y_nand[0]=1; Y_nand[1]=1;
    Y_nor[0]=1; Y_nor[1]=0;                                     #10 // 000 -10 ns bekle
    X=1;                                                         #10 //010
    X=0; Y_and[1]=1; Y_or[0]=1; Y_nand[1]=0; Y_nor[0]=0;       #10 //100
    X=1;                                                         #10 //110
    $stop; // simulasyonu durdur
end
endmodule

```

 <p>2x1 mux kullanılarak AND kapısı tasarımı</p>	 <p>2x1 mux kullanılarak OR kapısı tasarımı</p>
 <p>2x1 mux kullanılarak NAND kapısı tasarımı</p>	 <p>2x1 mux kullanılarak NOR kapısı tasarımı</p>

Tablo 3. Temel Kapıların 2x1 MuX kullanılarak tasarlanması

X seçmesi ve Y giriş değerleri, Tablo 3’de gösterilen temel kapıların doğruluk tablosu kullanılarak çıkarılmış 2x1 MuX girişlerinin değerlerine göre atanmıştır. Buna göre; topraklama olarak gösterilen yerlere 0 değeri, VCC ve 1 değeri gösterilen yerlere 1 değeri, A olarak gösterilen yerlere sırasıyla 0 ve 1 değerleri, A’ olarak gösterilen yerlere sırasıyla 1 ve 0 değerleri verilmiştir. Simülasyon sonucunda çıkan zamanlama diyagramı Şekil 4’ de gösterilmiştir.



Şekil 4. $2^N \times 1$ Çoğullayıcı ile Temel Lojik Kapılar Yapıldığında Zamanlama Diyagramı

c) $2^N \times 1$ MUX ile Donanım Tanımlama Dili Kullanarak Büyük Çoğullayıcıların Tasarımı

Problem 1 'deki $2^N \times 1$ MUX ile DTD kullanarak hiyerarşik düzende 4×1 , 8×1 , 16×1 MUX tasarlanmıştır. Yazılan kod aşağıdaki gibidir:

```
/* lab3_g7_p2.sv
* Hazırlayanlar: Umut Mehmet ERDEM - Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 - Problem 2
*  $2^N \times 1$  Çoğullayıcı ile Büyük Çoğullayıcıların Tasarımı
*/
`timescale 1ns/1ps
module lab3_g7_p2(
    input logic [1:0] S_mux4,
    input logic [2:0] S_mux8,
    input logic [3:0] Y_mux4, S_mux16,
    input logic [7:0] Y_mux8,
    input logic [15:0] Y16,
    output logic Q_mux4, Q_mux8, Q_mux16
);
endmodule
```

Ana modül içerisinde tanımlanan S seçme parametreleri, Y giriş parametreleri, Q çıkış parametreleri Testbench dosyasında kullanılmak üzere oluşturduğumuz Mux modülleri için tanımlanmıştır.

```

module mux4(
    input logic [3:0] P,
    input logic [1:0] S,
    output logic Q
);
    logic F1, F2;
    mux2 mux2_1(S[0], P[1:0], F1);
    mux2 mux2_2(S[0], P[3:2], F2);
    mux2 mux2_Q(S[1], {F1, F2}, Q);
endmodule

module mux8(
    input logic [7:0] P,
    input logic [2:0] S,
    output logic Q
);
    logic F1, F2;
    mux4 mux4_1(P[3:0], S[2:1], F1);
    mux4 mux4_2(P[7:4], S[2:1], F2);
    mux2 mux2_Q(S[0], {F1, F2}, Q);
endmodule

module mux16(
    input logic [15:0] P,
    input logic [3:0] S,
    output logic Q
);
    logic F1, F2;
    mux8 mux8_1(P[7:0], S[3:1], F1);
    mux8 mux8_2(P[15:8], S[3:1], F2);
    mux2 mux2_Q(S[0], {F1, F2}, Q);
endmodule

```

4x1 MuX tasarlamak için öncelikle “mux4” modülü; 4 bitlik P giriş parametresi, 2 bitlik S parametresi, 1 bitlik Q çıkış parametresi tanımlanarak oluşturulmuştur. 2 tane “mux2” modülü; “mux2_1” ve “mux2_2” olarak ilk aşamada 0. seçme biti verilerek P giriş bitlerinin 0, 1 ve 2, 3 bitleri olarak sırasıyla 2x1 MuX'lara parametre olarak verilmesi ile oluşturulmuştur. Çıkış bitleri F1 ve F2 olarak tanımlanmış değişkenlerde tutulmuş ve 1. Seçme biti verilerek kontrol edilecek son 2x1 MuX(“mux2_Q”) 'a sokulmuştur. Benzer hiyerarşik yapı 8x1 MuX tanımlanırken “mux8” modülünde yapılmıştır. Bu sefer 8 bitlik P giriş parametresi, 3 bitlik S parametresi, 1 bitlik Q çıkış parametresi tanımlanmıştır. Başta tanımladığımız mux4 modülü kullanarak P giriş bitleri sırasıyla 0-3 bitleri ilk “mux4” modülüne(“mux4_1”) girecek şekilde, 4-7 bitleri ikinci “mux4” modülüne(“mux4_2”) girecek şekilde ayarlanmıştır. 4x1 MuX modüllerine 1-2 seçme bitleri verilmiş ve modül içerisinde tanımlanmış olan F1, F2 değişkenlerinde çıkış değerleri tutulmuştur. Tutulan 2 tane çıkış bitleri “mux2” modülü ile tanımlanmış “mux2_Q” değişkenine, seçme biti 0. Bit olacak şekilde girdi olarak verilmiş ve Q çıkış değeri olarak çıktısı alınmıştır. “mux16” modülü tanımlanırken de aynı işlemler; 16 bitlik P giriş parametresi, 4 bitlik S giriş parametresi, 1 bitlik çıkış parametresi olacak şekilde “mux8” modülü ile tanımlanan “mux8_1” ve “mux8_2” modüllerine sırasıyla 0-7 bitleri ve 8-15 bitleri verilerek seçme bitleri ilk adımda 1-3 bitleri olacak şekilde verilmiştir. 8x1 MuX modüllerinden çıkan 2 tane çıkış bit değerleri(F1 ve F2) 2x1 mux modülüne seçme biti 0. bit olacak şekilde sokularak Q çıkış değeri alınmıştır.

```

/* tb_lab2_g7_p2.sv
* Hazırlayanlar: Umut Mehmet ERDEM - Emre TANER
* Notlar: ELM235 2023 Bahar Lab2 - Problem 2 Testbench
*  $2^N \times 1$  Çoğullayıcı ile Büyük Çoğullayıcıların Tasarımı
* Bütün olası girişlere göre çıkış gözlemlenir.
*/
// Zaman birimi ve simülasyon çözünürlüğü
`timescale 1ns/1ps
module tb_lab3_g7_p2();
// Test tezgahlarında port bulunmaz
    logic [3:0] S_mux16;
    logic [15:0] Y16;
    logic Q_mux16;
// Test edilecek modülün yaratımı ve port bağlantılarının yapılması
// dut = device under test
    lab3_g7_p2 dut0(S_mux16, Y16, Q_mux16);
    mux16 mux16_1(Y16[15:0], S_mux16[3:0], Q_mux16);
// Bu kısımda sinyaller test edilen devreye sıralı olarak uygulanır.
// Sonuçlar test edilen devre çıkışlarında gözlemlenebilir.
initial begin
    S_mux16[3]=0; S_mux16[2]=1; S_mux16[1]=0; S_mux16[0]=1; Y16[15]=1; Y16[14]=1;
Y16[13]=1; Y16[12]=0; Y16[11]=0; Y16[10]=0; Y16[9]=1; Y16[8]=0; Y16[7]=1; Y16[6]=0;
Y16[5]=1; Y16[4]=0; Y16[3]=0; Y16[2]=0; Y16[1]=1; Y16[0]=0; #10
    S_mux16[3]=0; S_mux16[2]=1; S_mux16[1]=1; S_mux16[0]=0; Y16[15]=0; Y16[14]=1;
Y16[13]=0; Y16[12]=0; Y16[11]=1; Y16[10]=1; Y16[9]=1; Y16[8]=0; Y16[7]=0; Y16[6]=1;
Y16[5]=0; Y16[4]=1; Y16[3]=0; Y16[2]=1; Y16[1]=0; Y16[0]=1; #10
    S_mux16[3]=1; S_mux16[2]=1; S_mux16[1]=0; S_mux16[0]=0; Y16[15]=0; Y16[14]=0;
Y16[13]=0; Y16[12]=1; Y16[11]=0; Y16[10]=1; Y16[9]=0; Y16[8]=0; Y16[7]=0; Y16[6]=1;
Y16[5]=0; Y16[4]=0; Y16[3]=1; Y16[2]=0; Y16[1]=0; Y16[0]=0; #10
    S_mux16[3]=1; S_mux16[2]=0; S_mux16[1]=0; S_mux16[0]=1; Y16[15]=0; Y16[14]=0;
Y16[13]=0; Y16[12]=0; Y16[11]=0; Y16[10]=0; Y16[9]=0; Y16[8]=0; Y16[7]=0; Y16[6]=1;
Y16[5]=0; Y16[4]=0; Y16[3]=1; Y16[2]=0; Y16[1]=0; Y16[0]=1; #10
    $stop;
end
endmodule
/*
logic [1:0] S_mux4;
logic [3:0] Y_mux4;
logic Q_mux4;
lab3_g7_p2 dut0(S_mux4, Y_mux4, Q_mux4);
mux4 mux4_1(Y_mux4[3:0], S_mux4[1:0], Q_mux4);
initial begin
    S_mux4[1]=0; S_mux4[0]=0; Y_mux4[3]=0; Y_mux4[2]=0; Y_mux4[1]=0; Y_mux4[0]=0; #10
    S_mux4[1]=0; S_mux4[0]=0; Y_mux4[3]=0; Y_mux4[2]=0; Y_mux4[1]=0; Y_mux4[0]=1; #10
    S_mux4[1]=0; S_mux4[0]=1; Y_mux4[3]=0; Y_mux4[2]=0; Y_mux4[1]=0; Y_mux4[0]=0; #10
    S_mux4[1]=0; S_mux4[0]=1; Y_mux4[3]=0; Y_mux4[2]=0; Y_mux4[1]=1; Y_mux4[0]=0; #10
    S_mux4[1]=1; S_mux4[0]=0; Y_mux4[3]=0; Y_mux4[2]=0; Y_mux4[1]=0; Y_mux4[0]=0; #10
    S_mux4[1]=1; S_mux4[0]=0; Y_mux4[3]=0; Y_mux4[2]=1; Y_mux4[1]=0; Y_mux4[0]=0; #10
    S_mux4[1]=1; S_mux4[0]=1; Y_mux4[3]=0; Y_mux4[2]=0; Y_mux4[1]=0; Y_mux4[0]=0; #10
    S_mux4[1]=1; S_mux4[0]=1; Y_mux4[3]=1; Y_mux4[2]=0; Y_mux4[1]=0; Y_mux4[0]=0; #10
    $stop;
end
endmodule
*/

```



Şekil 5. $2^N \times 1$ Çoğullayıcı ile 16×1 Yapıldığı Zamanlama Diyagramı

2.2.2. Sonuçların Yorumu

2×1 Çoğullayıcı kullanarak temel lojik kapıların ve büyük çoğullayıcıların tasarımı yapılması öğrenilmiştir. Temel lojik kapılar oluştururken istenilen lojik kapiya göre 2×1 çoğullayıcının girişine değerler atandığı gözlemlenmiştir. 4×1 Çoğullayıcı için 3 adet 2×1 çoğullayıcı, 8×1 çoğullayıcı için 2 adet 4×1 ve 1 adet 2×1 çoğullayıcı, 16×1 çoğullayıcı için 2 adet 8×1 ve 1 adet 2×1 çoğullayıcı kullanıldığı gözlemlenmiştir.

2.3. Problem III - 2×4 Kod Çözücü (Decoder) Tasarımı

2.3.1. Teorik Araştırma

Decoder'ın tanımı şu şekildedir:

Kod çözücü (decoder), temel olarak kodlanmış verinin ilk halini tekrar elde etmek için kullanılmaktadır. Sayısal elektronikte kod çözücülerin basit mantığı, kodlanmış çoklu giriş kod çözücüye verilmekte ve çıkış olarak da farklı şekilde kodlanmış çoklu çıkış alınmaktadır.

2.3.2. Deneyin Yapılışı

a) 2×4 Decoder Devresini Donanım Tanımlama Dili Kullanarak Gerçeklenmesi

2×4 Decoder devresini DTD kullanarak kapı düzeyinde sadece NAND kapıları kullanarak tasarlanıldı. Yazılan kod aşağıdaki gibidir.

```

/* lab3_g7_p3.sv
* Hazırlayanlar: Umut Mehmet ERDEM - Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 - Problem 3
* 2 x 4 Decoder Devresini DTD Dili Kullanarak Gerçeklenmesi
*/
`timescale 1ns/1ps
module lab3_g7_p3(
    input logic S,
    input logic [1:0] I,
    output logic [3:0] Q
);
    decoder2x4 decoder2x4(S, I[1:0], Q[3:0]);
endmodule

module decoder2x4(
    input logic S,
    input logic [1:0] I,
    output logic [3:0] Q
);
    assign Q[0] = S&~((~(I[1]&I[1])&~(I[0]&I[0]))&~((I[1]&I[1])&~(I[0]&I[0])));
    assign Q[1] = S&~((~(I[1]&I[1])&I[0])&~((I[1]&I[1])&I[0]));
    assign Q[2] = S&~((~(I[0]&I[0])&I[1])&~((I[0]&I[0])&I[1]));
    assign Q[3] = S&~((I[0]&I[1])&~(I[0]&I[1]));
endmodule

```

Yazılan bu kodda S enable bitini, I decoderin girişlerini, Q decoderin çıkışlarını tanımlamaktadır. Oluşturulmuş “decoder2x4” modülünde giriş bitlerinin 4 farklı çıkış kombinasyonları I1’I0’, I1’I0, I1I0’, I1I0 olarak NAND kapıları ile olacak şekilde denklemleri Q çıkış bitlerine “assign” ile atanmıştır. Enable biti NAND kapıları ile oluşturulmuş denklemlere AND kapısı ile bağlanmıştır bunun nedeni enable bitinin 0 olması durumunda bitlerin sıfırlanması içindir. Ana modül içerisinde oluşturulmuş “decoder2x4” için “decoder2x4” isminde modül tanımlanmıştır ve parametreler verilmiştir.

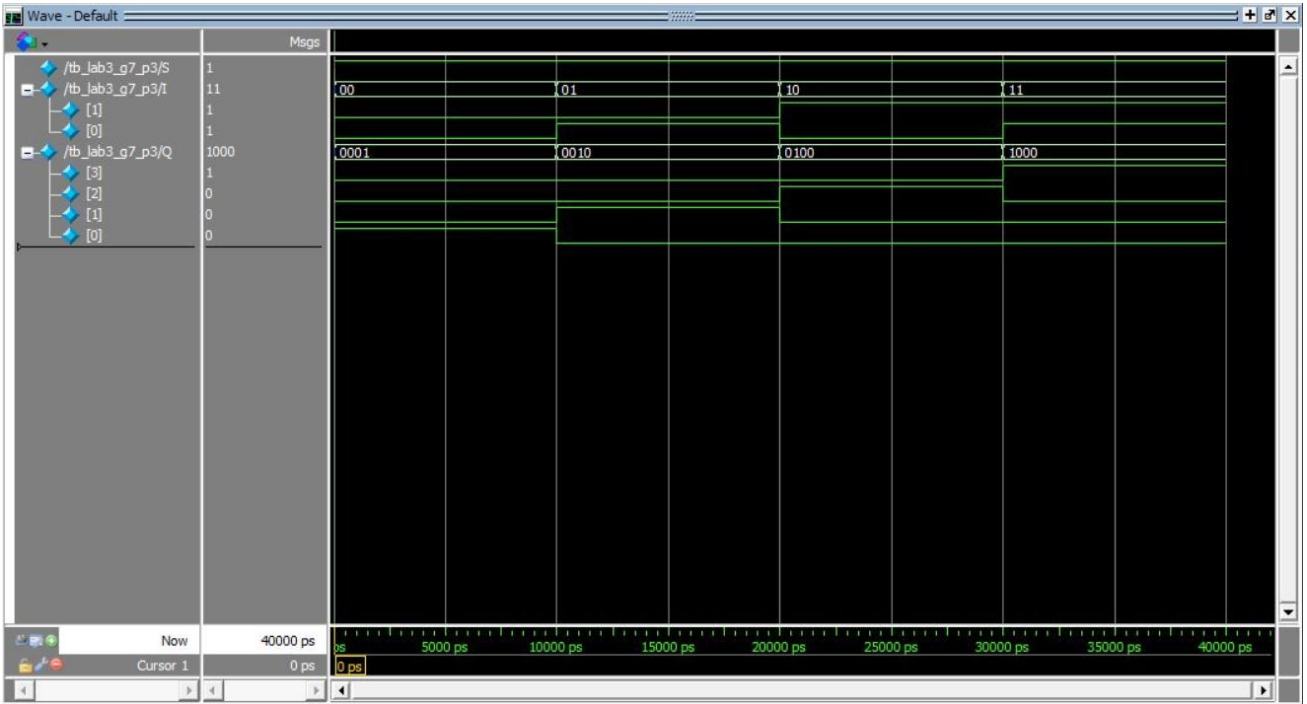
b) Tasarlanılan DTD Modelinin Simüle Edilmesi ve Zamanlama Diyagramı

Testbench dosyası içerisinde simüle etmek için dut0 kısmında parametreler girilmiş ve enable portu 1 olacak şekilde I giriş portuna doğruluk tablosuna göre değerler atama yapılmıştır ve Şekil 6’ da gösterilen zamanlama diyagramı elde edilmiştir.

```

/* tb_lab3_g7_p3.sv
* Hazırlayanlar: Umut Mehmet ERDEM - Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 - Problem 3 Testbench
* 2 x 4 Decoder Devresini DTD Dili Kullanarak Gerçeklenmesi
* Bütün olası girişlere göre çıkış gözlemlenir.
*/
// Zaman birimi ve simülasyon çözünürlüğü
`timescale 1ns/1ps
module tb_lab3_g7_p3();
// Test tezgahlarında port bulunmaz
    logic S;
    logic [1:0]I; // test tezgahi giriş sinyal tanımları
    logic [3:0]Q; // test tezgahi çıkış sinyal tanımları
// Test edilecek modülün yaratımı ve port bağlantılarının yapılması
// dut = device under test
    lab3_g7_p3 dut0(S, I, Q);
// Bu kısımda sinyaller test edilen devreye sıralı olarak uygulanır.
// Sonuçlar test edilen devre çıkışlarında gözlemlenebilir.
initial begin
    S=1; I[1]=0; I[0]=0;          #10 -10 ns bekle
    I[1]=0; I[0]=1;              #10
    I[1]=1; I[0]=0;              #10
    I[1]=1; I[0]=1;              #10
    $stop; // simülasyonu durdur
end
endmodule

```



Şekil 6. 2 x 4 Decoder'in Zamanlama Diyagramı

2.3.3. Sonuçların Yorumu

2 x 4 decoder oluştururken iki giriş, bir enable ve dört çıkış portu olarak tasarlanmıştır. Çıkışlar girişlerin bir kombinasyonu şeklinde tanımlanmıştır. Problem 3'te istenilen decoder başarılı bir şekilde tasarlanmış ve rapora eklenmiştir.

2.4. Problem IV- 4×16 Kod Çözücü (Decoder) Hiyerarşik Tasarımı

2.4.1. Deneyin Yapılışı

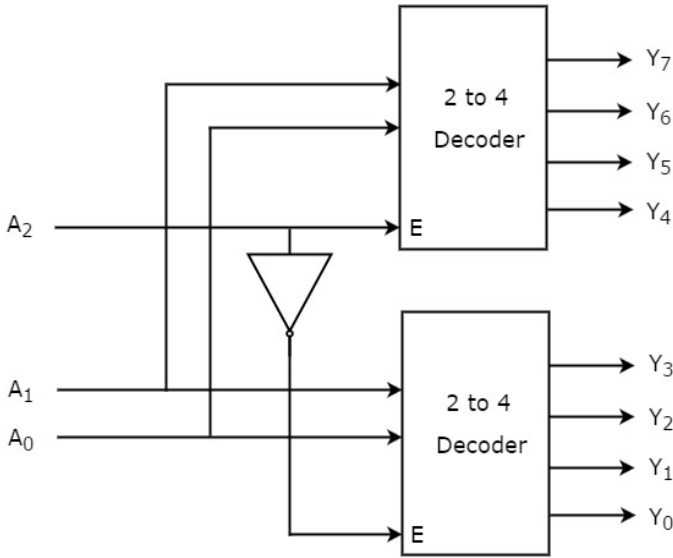
c) 2×4 Decoder ile Donanım Tanımlama Dili Kullanarak 3×8 ve 4×16 Decoder Yapımı

Problem 3’ te tanımladığımız 2×4 decoder devresini kullanarak 3×8 ve 4×16 decoder tasarlanmıştır. Yazılan kod aşağıdaki gibidir:

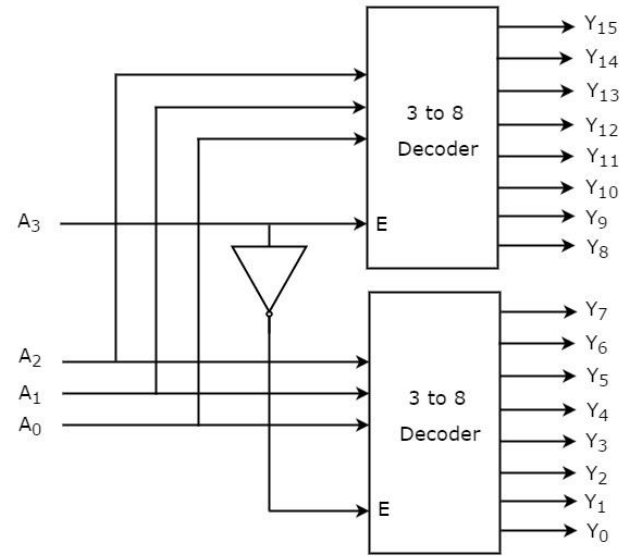
```
/* lab3_g7_p4.sv
* Hazırlayanlar: Umut Mehmet ERDEM – Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 – Problem 4
*  $2 \times 4$  Decoder Devresi ile DTD Dili Kullanarak  $3 \times 8$  ve  $4 \times 16$  Gerçeklenmesi
*/
module lab3_g7_p4(
    input logic S,
    input logic [2:0] I_3,
    input logic [3:0] I_4,
    output logic [7:0] I_8,
    output logic [15:0] Q_16
);
endmodule

module decoder3x8(
    input logic S,
    input logic [2:0] I,
    output logic [7:0] Q
);
    wire A;
    logic [3:0] F1, F2;
    assign A = ~(I[2]&I[2]);
    decoder2x4 decoder2x4_1(A, I[1:0], F1[3:0]);
    assign Q[3]= S&F1[3];
    assign Q[2]= S&F1[2];
    assign Q[1]= S&F1[1];
    assign Q[0]= S&F1[0];
    decoder2x4 decoder2x4_2(I[2], I[1:0], F2[3:0]);
    assign Q[7]= S&F2[3];
    assign Q[6]= S&F2[2];
    assign Q[5]= S&F2[1];
    assign Q[4]= S&F2[0];
endmodule

module decoder4x16(
    input logic [3:0] I,
    output logic [15:0] Q
);
    wire A;
    assign A = ~(I[3]&I[3]);
    decoder3x8 decoder3x8_1(A, I[2:0], Q[7:0]);
    decoder3x8 decoder3x8_2(I[3], I[2:0], Q[15:8]);
endmodule
```



Şekil 7. 2x4 Decoder kullanarak 3x8 Decoder Yapımı



Şekil 8. 3x8 Decoder kullanarak 4x16 Decoder Yapımı

3x8 decoder tasarlamak için “decoder3x8” modülü içerisinde problem 3’ te oluşturmuş olduğumuz 2x4 decoder modülü kullanılmıştır. “decoder2x4” modülü içerisinde bulunan enable port parametresi olarak 3x8 decoder da bulunan 3.giriş portu verilmiştir. Bunun haricinde 3x8 decoder kullanılarak 4x16 decoder elde etmek istediğimizden 3x8 decoder üzerinde enable portu verilmiştir. “decoder2x4_1” modülünün ilk parametresine Şekil 7’ de gözüktüğü gibi enable portu olarak 3.portun tersi verilmiştir. Bu nedenle “decoder3x8” modülü içerisinde bir A değişkeni tanımlanmış ve NAND kapısı şeklinde I değişkeninin 3. bitinin tersi yani 3. giriş portu tersi atanmıştır. Her iki “decoder2x4” modülüne de giriş parametresi olarak 3x8 decoder da bulunan 3 giriş portuna yani 3 bite sahip I değişkeninin 1-2 bitleri verilmiştir. “decoder2x4_2” modülüne enable portu olarak 3. Portun kendisi yani I değişkeninin 2.biti verilmiştir. 4 bitlik çıkış değerlerini tutması için 4 bitlik F1 ve F2 değişkenleri tanımlanmış ve her iki değişkeninin bulunan bit değerleri 3x8 decoder da tanımlanan S enable değişkeni ile çarpılarak Q çıkış değişkenine ataması yapılmıştır. 4x16 decoder da benzer şekilde yapılmıştır fakat enable portu tanımlanmamıştır. 3x8 decoder kullanılarak enable parametresi için 4.giriş portu yani I değişkeninin 4.biti iki farklı “decoder3x8” modülüne tersi ve kendisi olmak üzere Şekil 8’ deki verilmiş, çıkış parametrelerine ise direkt “decoder4x16” modülünde tanımladığımız Q değişkeninin bitleri sırasıyla 0-7 ve 8-15 olacak şekilde atanmıştır. Giriş portu olarak I değişkeninin 0-2 indeksine sahip bitleri verilmiştir. Bu şekilde 4x16 modülü ve ana modül içerisinde testbench dosyasında parametre olarak verilecek parametreler tanımlanmıştır. Testbench dosyası içerisinde simüle edebilmemiz için dut0 belirtilmiş ve parametreler girilmiştir. 4x16 decoder üzerinde bulunan giriş portlarının doğruluk tablosuna göre ataması yapılmış ve zamanlama diyagramı Şekil 9’ da gösterilmiştir.

d) Tasarlanılan DTD Modelinin Simüle Edilmesi ve Zamanlama Diyagramı

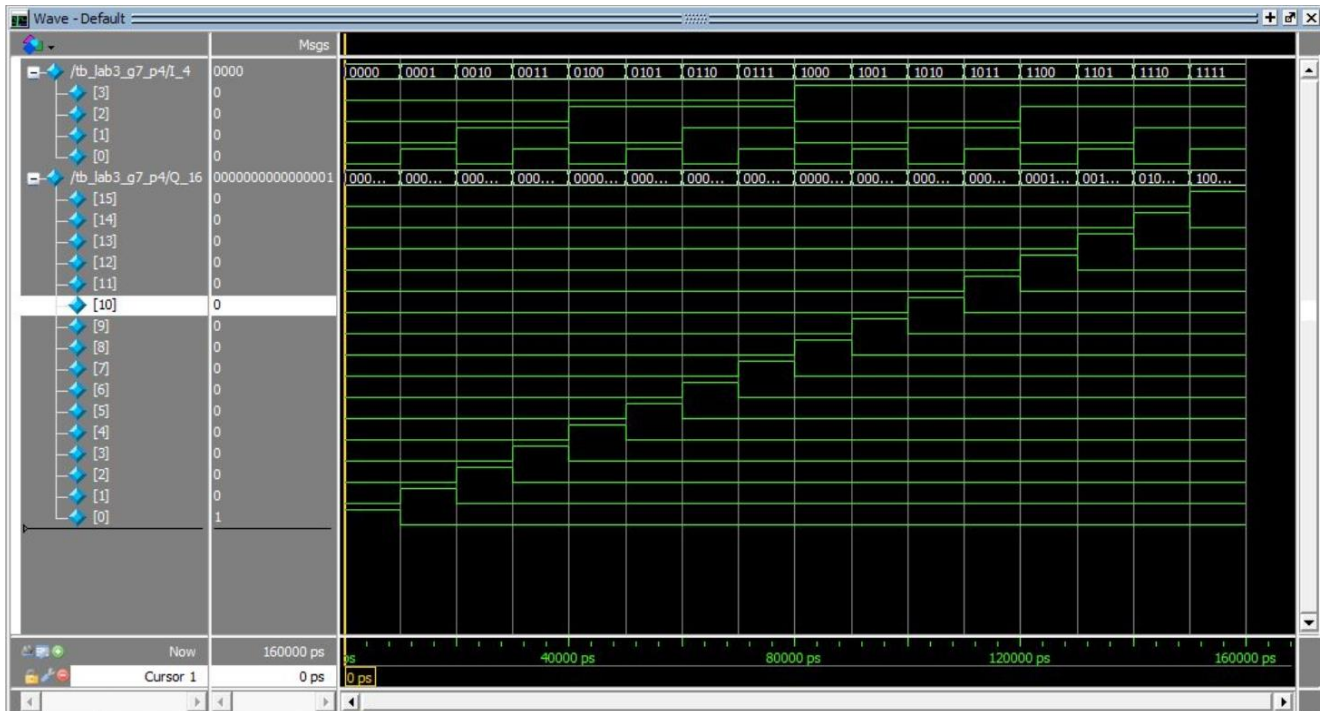
```
/* tb_lab3_g7_p4.sv
* Hazırlayanlar: Umut Mehmet ERDEM - Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 - Problem 4 Testbench
* 2 x 4 Decoder Devresini DTD Dili Kullanarak 3 x 8 ve 4 x 16 Gerçeklenmesi
* Bütün olası girişlere göre çıkış gözlemlenir.
*/
// Zaman birimi ve simülasyon çözünürlüğü
`timescale 1ns/1ps
module tb_lab3_g7_p4();
// Test tezgahlarında port bulunmaz
    logic S;
    logic [2:0]I_3; // test tezgahi giriş sinyal tanımları
    logic [3:0]I_4;
    logic [7:0]Q_8; // test tezgahi çıkış sinyal tanımları
    logic [15:0] Q_16;
// Test edilecek modülün yaratımı ve port bağlantılarının yapılması
// dut = device under test
    lab3_g7_p4 dut0(S, I_4, Q_16);
    decoder4x16 decoder4x16(I_4[3:0], Q_16[15:0]);
// Bu kısımda sinyaller test edilen devreye sıralı olarak uygulanır.
// Sonuçlar test edilen devre çıkışlarında gözlemlenebilir.
initial begin
    I_4[3]=0; I_4[2]=0; I_4[1]=0; I_4[0]=0; #10 -10 ns bekle
    I_4[3]=0; I_4[2]=0; I_4[1]=0; I_4[0]=1; #10
    I_4[3]=0; I_4[2]=0; I_4[1]=1; I_4[0]=0; #10
    I_4[3]=0; I_4[2]=0; I_4[1]=1; I_4[0]=1; #10
    I_4[3]=0; I_4[2]=1; I_4[1]=0; I_4[0]=0; #10
    I_4[3]=0; I_4[2]=1; I_4[1]=0; I_4[0]=1; #10
    I_4[3]=0; I_4[2]=1; I_4[1]=1; I_4[0]=0; #10
    I_4[3]=0; I_4[2]=1; I_4[1]=1; I_4[0]=1; #10
    I_4[3]=1; I_4[2]=0; I_4[1]=0; I_4[0]=0; #10
    I_4[3]=1; I_4[2]=0; I_4[1]=0; I_4[0]=1; #10
    I_4[3]=1; I_4[2]=0; I_4[1]=1; I_4[0]=0; #10
    I_4[3]=1; I_4[2]=0; I_4[1]=1; I_4[0]=1; #10
    I_4[3]=1; I_4[2]=1; I_4[1]=0; I_4[0]=0; #10
    I_4[3]=1; I_4[2]=1; I_4[1]=0; I_4[0]=1; #10
    I_4[3]=1; I_4[2]=1; I_4[1]=1; I_4[0]=0; #10
    I_4[3]=1; I_4[2]=1; I_4[1]=1; I_4[0]=1; #10
    $stop; // simülasyonu durdur
end
endmodule
```

```

/*
lab3_g7_p4 dut0(S, I_3, Q_8);

decoder3x8 decoder3x8(S, I_3[2:0], Q_8[7:0]);
initial begin
    S=1; I_3[2]=0; I_3[1]=0; I_3[0]=0; #10
    I_3[2]=0; I_3[1]=0; I_3[0]=1; #10
    I_3[2]=0; I_3[1]=1; I_3[0]=0; #10
    I_3[2]=0; I_3[1]=1; I_3[0]=1; #10
    I_3[2]=1; I_3[1]=0; I_3[0]=0; #10
    I_3[2]=1; I_3[1]=0; I_3[0]=1; #10
    I_3[2]=1; I_3[1]=1; I_3[0]=0; #10
    I_3[2]=1; I_3[1]=1; I_3[0]=1; #10
    $stop;
end
endmodule
*/

```



Şekil 9. 4×16 Decoder'in Zamanlama Diyagramı

2.4.2. Sonuçların Yorumu

2×4 Decoder kullanarak 3×8 ve 4×16 decoder tasarımı yapılması öğrenilmiştir. 3×8 decoder için 2 adet 2×4 decoder, 4×16 decoder için 2 adet 3×8 decoder kullanıldığı gözlemlenmiştir.

2.5. Problem V - Çoğullayıcı (MUX) ve Çözücüler (Decoder) ile Birleşik Devre Tasarımı

2.5.1. Deneyin Yapılışı

a) Problem 5'te Verilen F_1 ve F_2 Fonksiyonları MUX ve Decoder Devrelerini Kullanarak DTD ile Yapısal Şekilde Tasarlanması

Verilen F_1 denklemi minterm olarak verildiğinden direk olarak decoder kullanılmıştır. F_2 denklemi maxterm olarak verildiğinden ilk önce minterm yapıp daha sonra decoder kullanılmıştır. Yazılan kodlar aşağıdaki gibidir:

```

/* lab3_g7_p5.sv
* Hazırlayanlar: Umut Mehmet ERDEM – Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 – Problem 5
* MUX ve Decoder ile Birleşik Devre Tasarımı
*/
`timescale 1ns/1ps
module lab3_g7_p5(
    input logic [3:0] I,
    output logic [15:0] Q1, Q2,
    output logic F1,F2

);
    decoder4x16 dec4_1(I[3:0], Q1[15:0]);
    decoder4x16 dec4_2(I[3:0], Q2[15:0]);
    assign F1 = Q1[1]+Q1[3]+Q1[7]+Q1[11]+Q1[12]+Q1[13]+Q1[15];
    assign F2 = Q2[0]+Q2[1]+Q2[6]+Q2[8]+Q2[9]+Q2[10]+Q2[11]+Q2[12]+Q2[13]+Q2[14]+Q2[15];
endmodule

```

Problem 4’ de tanımladığımız “decoder4x16” modüllerine 4 giriş portlarını içeren 4 bitlik değişken, Q1 ve Q2 çıkış portlarını içeren 16 bitlik değişkenleri parametre olarak verilmiştir. Verilen F1 fonksiyonlarındaki minterm çıkışları toplanmış ve F1 değişkenine atanmış, F2 fonksiyonlarındaki maxterm ifadeler minterm yapılarak F2 değişkenine atanmıştır. Testbench dosyasında modüllerine 4 giriş portlarını içeren 4 bitlik değişkenin değerleri doğruluk tablosuna göre ataması yapılmış ve F1, F2 için Şekil 10’ da zamanlama diyagramı çizilmiştir.

b) Tasarılan DTD Modelinin Simüle Edilmesi ve Zamanlama Diyagramı

```

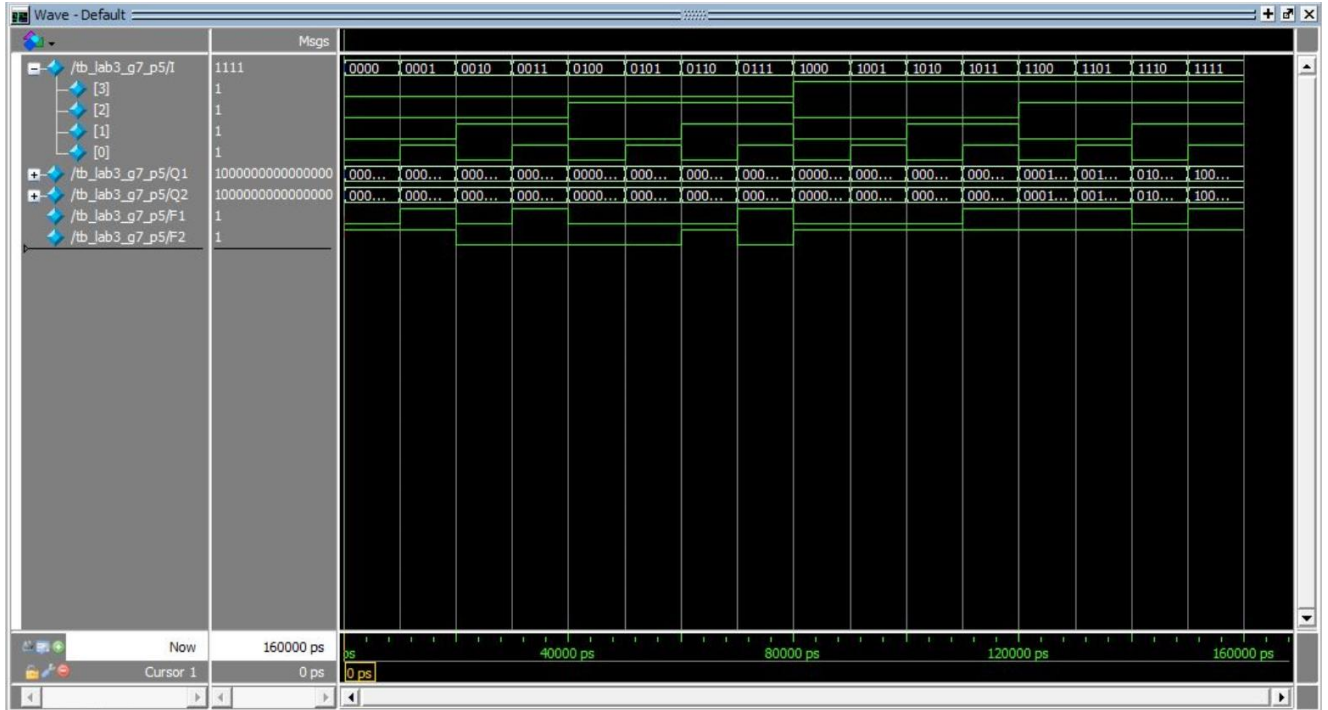
/* tb_lab3_g7_p5.sv
* Hazırlayanlar: Umut Mehmet ERDEM – Emre TANER
* Notlar: ELM235 2023 Bahar Lab3 – Problem 5 Testbench
* MUX ve Decoder ile Birleşik Devre Tasarımı
* Bütün olası girişlere göre çıkış gözlemlenir.
*/
// Zaman birimi ve simülasyon çözünürlüğü
`timescale 1ns/1ps
module tb_lab3_g7_p5();
    // Test tezgahlarında port bulunmaz
    logic [3:0]I; // test tezgahi giris sinyal tanimlari
    logic [15:0]Q1,Q2; // test tezgahi cikis sinyal tanimlari
    logic F1,F2;
    // Test edilecek modulun yaratimi ve port baglantilarinin yapılması
    // dut = device under test
    lab3_g7_p5 dut0(I,Q1,Q2,F1,F2);
    // Bu kismda sinyaller test edilen devreye sirali olarak uygulanir.
    // Sonuclar test edilen devre cikislarinda gozlenebilir.

```

```

initial begin
    I[3]=0; I[2]=0; I[1]=0; I[0]=0; #10 -10 ns bekle
    I[3]=0; I[2]=0; I[1]=0; I[0]=1; #10
    I[3]=0; I[2]=0; I[1]=1; I[0]=0; #10
    I[3]=0; I[2]=0; I[1]=1; I[0]=1; #10
    I[3]=0; I[2]=1; I[1]=0; I[0]=0; #10
    I[3]=0; I[2]=1; I[1]=0; I[0]=1; #10
    I[3]=0; I[2]=1; I[1]=1; I[0]=0; #10
    I[3]=0; I[2]=1; I[1]=1; I[0]=1; #10
    I[3]=1; I[2]=0; I[1]=0; I[0]=0; #10
    I[3]=1; I[2]=0; I[1]=0; I[0]=1; #10
    I[3]=1; I[2]=0; I[1]=1; I[0]=0; #10
    I[3]=1; I[2]=0; I[1]=1; I[0]=1; #10
    I[3]=1; I[2]=1; I[1]=0; I[0]=0; #10
    I[3]=1; I[2]=1; I[1]=0; I[0]=1; #10
    I[3]=1; I[2]=1; I[1]=1; I[0]=0; #10
    I[3]=1; I[2]=1; I[1]=1; I[0]=1; #10
    $stop; // simulasyonu durdur
end
endmodule

```



Şekil 10. Problem 5'te Verilen F1 ve F2'nin Zamanlama Diyagramı

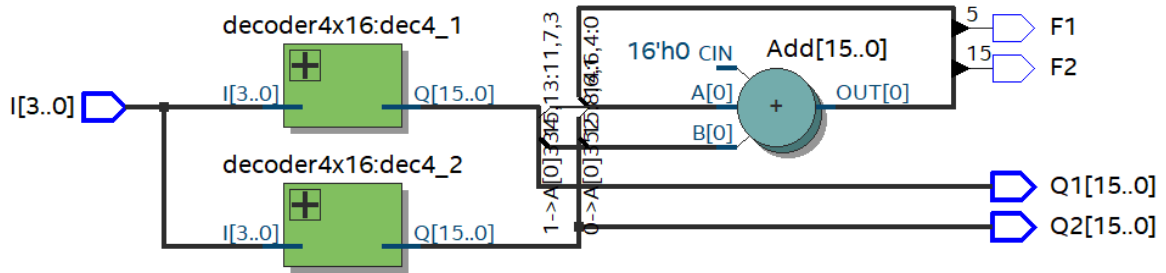
c)

Flow Summary	
<<Filter>>	
Flow Status	Successful - Thu May 11 23:58:15 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	lab3_g7_p5
Top-level Entity Name	lab3_g7_p5
Family	MAX 10
Device	10M08DAF484C8G
Timing Models	Final
Total logic elements	19 / 8,064 (< 1 %)
Total registers	0
Total pins	38 / 250 (15 %)
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 %)

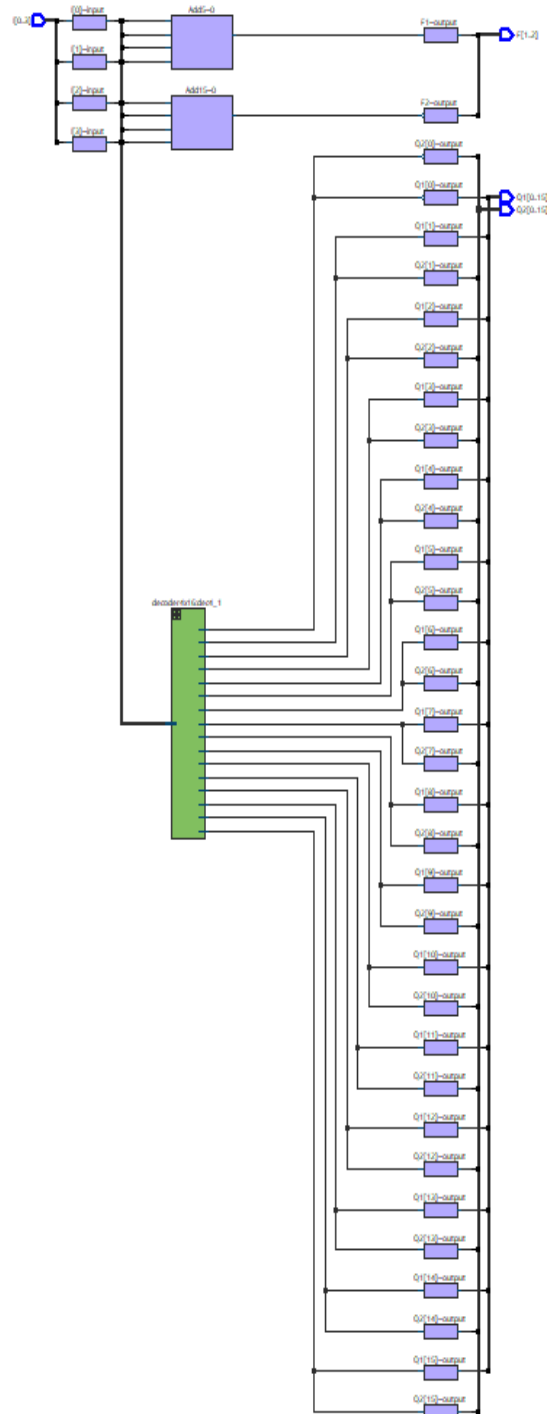
Şekil 11. Problem 5' in Analiz ve Sentez Özeti

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimated Total logic elements	18
2		
3	Total combinational functions	18
4	▼ Logic element usage by number of LUT inputs	
1	-- 4 input functions	18
2	-- 3 input functions	0
3	-- <=2 input functions	0
5		
6	▼ Logic elements by mode	
1	-- normal mode	18
2	-- arithmetic mode	0
7		
8	▼ Total registers	0
1	-- Dedicated logic registers	0
2	-- I/O registers	0
9		
10	I/O pins	38
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	I[1]...nput
15	Maximum fan-out	18
16	Total fan-out	144
17	Average fan-out	1.53

Şekil 12. Problem 5' in Analiz ve Sentez Kaynak Kullanımı Özeti



Şekil 13. Problem 5'in RTL Şeması



Şekil 14. Problem 5' in Eşleştirme Sonrası Teknoloji Şeması

Şekil 12 ve Şekil 14' de gözüktüğü gibi toplam 18 mantıksal hücre kullanılmıştır. 4 girişli decoder kullanıldığından 18 tane 4 girişli mantıksal hücre gözükmektedir. Toplam pin sayısı 38 olarak gözükmektedir. Şekil 13' teki RTL şemasında decoder üzerinden çıkan çıkış portlarından istenen portlar toplanarak F1 ve F2 fonksiyonlarını elde etmektedir.

d)

AB\CD	C'D'	C'D	CD	CD'
A'B'	0	1	1	0
A'B	0	0	1	0
AB	1	1	1	0
AB'	0	0	1	0

Tablo 4. F1 Denkleminin K-Map Yardımıyla Sadeleştirilmesi

$$F_1 = CD + A'B'D + ABC'$$

AB\CD	C'D'	C'D	CD	CD'
A'B'	1	1	0	0
A'B	0	0	0	1
AB	1	1	1	1
AB'	1	1	1	1

Tablo 5. F2 Denkleminin K-Map Yardımıyla Sadeleştirilmesi

$$F_2 = A + B'C' + BCD'$$

K-map üzerinden hesaplanan F1 fonksiyonu için toplam 10 devre elemanı kullanılmıştır. F2 fonksiyonu için ise 8 devre elemanı kullanılmıştır.

- e) Şekil 12, Şekil 13 ve Şekil 14 göz önüne alındığında K-map üzerinden yapılan işlemler sonucunda bulunun denkleme göre devre elemanı sayısı daha az ve daha ekonomik olacağı görülmüştür.

2.5.2. Sonuçların Yorumu

Problem 4' de oluşturmuş olduğumuz 4x16 decoder kullanarak verilmiş olan F1 ve F2 fonksiyonları decoder üzerindeki çıktı portlarına göre toplanmış ve buna göre bulunan zamanlama diyagramının doğru sinyaller verdiği görülmüştür. E şıkında K-map üzerinden bulunan en sade Boole cebri ifadesinin 4x16 decoder' a göre daha mantıklı olduğu görülmüştür.

3. Sonular ve Genel Yorumlar

Yapılan laboratuvar alıřmasında multiplexer ve decoder tanımları renilmiř, nerelerde ve ne řekilde kullanıldıkları anlařılmıřtır. 2x1 multiplexer ve 2x4 decoder kullanılarak hiyerarřik bir yapıda structural olarak adım adım eklemeler ile 4x1, 8x1, 16x1 multiplexer ve 3x8, 4x16 decoder elde edilmiřtir. Bunlara gre zamanlama diyagramları ıkarılmıř ve sinyallerin doėruluėu test edilmiřtir. Bunlara gre devrede kullanılacak elemanların karřılařtırması yapılmıř ve hangisinin daha ekonomik olacaėı grlmřtr. Structural tarzda kod yazma esnasında tanımlanan modllerin ana dosyada ana modl ierisinde parametre ataması yapılması sırasında sinyallerin zamanlama diyagramında gzkmemesi ancak testbench dosyasında parametre girildiėinde sinyallerin gzkmesi gibi birok yazılımsal hata alınmıř ve stesinden gelinmiřtir.

4. Referanslar

- [1] https://www.electronics-tutorials.ws/combinational/comb_5.html
- [2] https://www.sathyabama.ac.in/sites/default/files/course-material/2020-10/Unit2_0.pdf
- [3] <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c4/c4s2/c4s2v6/>
- [4] https://tr.wikipedia.org/wiki/Kod_%C3%A7%C3%B6z%C3%BCc%C3%BC
- [5] <https://www.electronicengineeringconcepts.com/2020/12/design-of-and-gate-using-multiplexer.html>
- [6] <https://www.electronicengineeringconcepts.com/2020/12/2-input-or-gate-using-21-multiplexer.html>
- [7] <https://www.electronicengineeringconcepts.com/2020/12/2-input-nand-gate-using-21-multiplexer.html>
- [8] <https://www.electronicengineeringconcepts.com/2020/12/2-input-nor-gate-using-21-multiplexer.html>
- [9] <https://tr.wikipedia.org/wiki/%C3%87oklay%C4%B1c%C4%B1>