GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

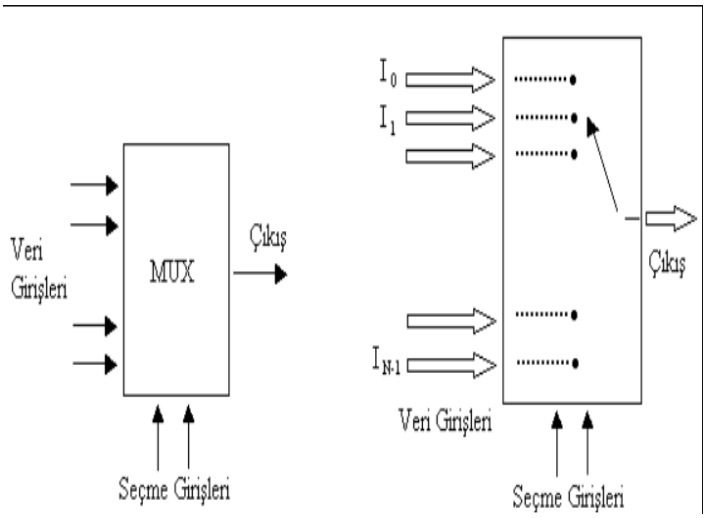
LAB 0x4 Deney Raporu

Kombinasyonel Devreler Tabanlı Tasarım.

Hazırlayanlar
1) 1901022038 – Selen Erdoğan
2) 1901022025–Ayşe Serra ŞİMŞEK

1. **Giriş**
2. **Problemler**

2.1. Problem I - 2x1 MUX devresini şematik düzeyde sadece NAND kapıları kullanarak tasarlayınız.



Şekil 1

2.1.1. Teorik Araştırma

Bir çok giriş hattından gelen bilgilerden birisini seçerek uygun çıkış hattına yönlendirilmesini sağlayan birleşik devrelere "çoklayıcı / veri seçici devreler" (multiplexer) denir ve ÇOĞ (MUX) sembolü ile gösterilir. Veri çoklayıcılar, orijinal isminden hareketle çoğu kere "multiplexer" olarak adlandırılmaktadır.

Şekil 1'de sembolü ve fonksiyon şeması görünen veri seçici devresinde girişteki bilgilerden uygun olanının seçilmesi işlemi seçme girişleri (select inputs) ile yapılır. Veri seçicilerde, 2n sayıdaki giriş hattından uygun olanı seçmek için "n" sayıda seçme hattına ihtiyaç vardır. Dijital olarak kontrol edilebilen çok pozisyonlu anahtar gibi işlem yapan veri seçiciler, seçme hattının girişlerindeki değere göre çıkışa aktarılacak giriş hattına karar verir.

2.1.2. Deneyin Yapılışı

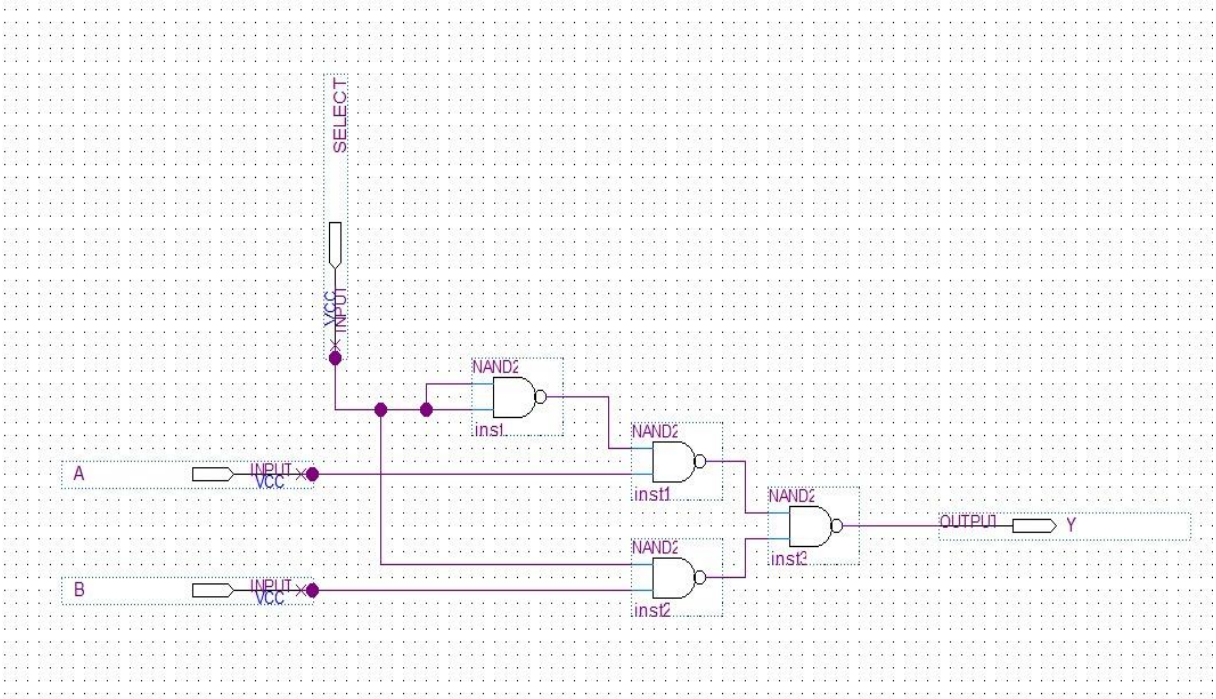
```
module MUX21_nand(
      A,
      B,
      SELECT,
      y
);


input wire A;
input wire B;
input wire SELECT;
output wire     y;

wire  SYNTHESIZED_WIRE_0;
wire  SYNTHESIZED_WIRE_1;
wire  SYNTHESIZED_WIRE_2;
assign     SYNTHESIZED_WIRE_0 = ~(SELECT & SELECT);
```

```
assign     SYNTHESIZED_WIRE_2 = ~(A & SYNTHESIZED_WIRE_0);
assign     SYNTHESIZED_WIRE_1 = ~(B & SELECT);
assign     y = ~(SYNTHESIZED_WIRE_1 & SYNTHESIZED_WIRE_2);
endmodule
```

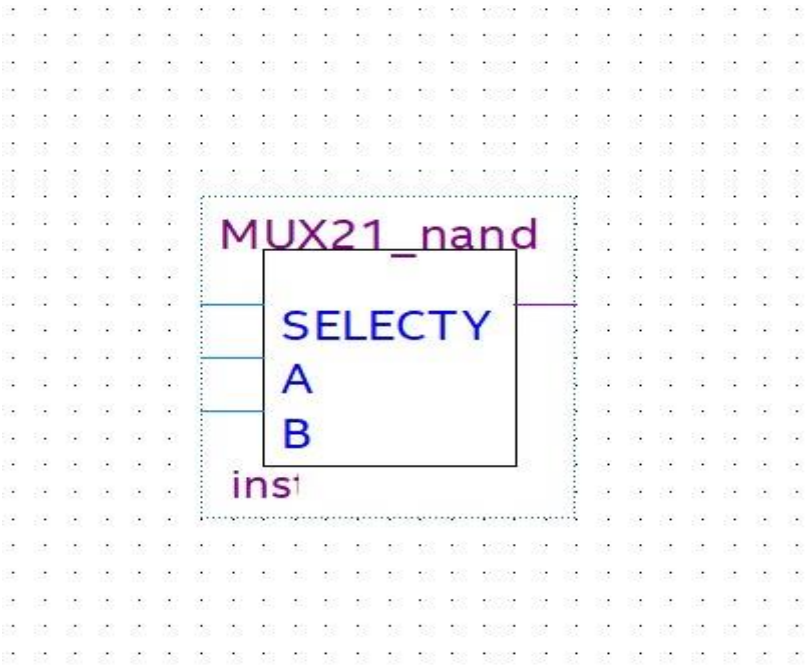
Tablo 1: Problem 1

```
`timescale 1ns/1ps
module tb_MUX21_nand();
logic a,b,s;
logic y;
MUX21_nand inst1(.A(a), .B(b), .SELECT(s), .y(y));
initial begin
a=0; b=0; s=1; #10;
a=0; b=1; s=1; #10
a=1; b=0; s=1; #10;
a=1; b=1; s=1; #10;
a=0; b=0; s=1; #10;
a=0; b=1; s=1; #10
a=1; b=0; s=1; #10;
a=1; b=1; s=1; #10;
#20;
$stop;
end
endmodule
```

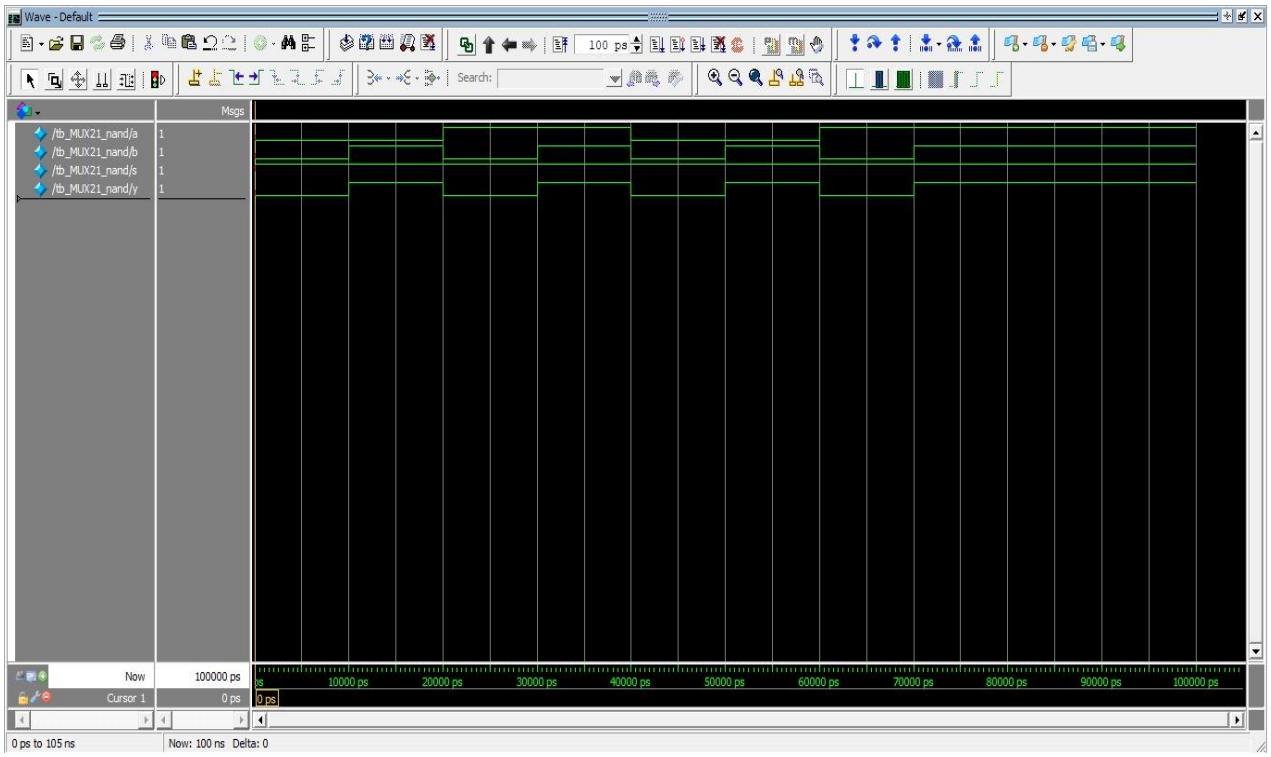
Tablo 2: Problem 1 Testbench



Şekil 2 Problem 1 Şematik Çizimi



Şekil 3 Problem 1 Şematik



Şekil 4 Problem 1 ModelSim Dalga Şeması

Problem 1 – SONUÇ : 2x1 MUX devresi şematik düzeyde sadece NAND kapıları kullanarak tasarlandı. Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemlendi. ModelSim dalga şeması elde edildi.

Problem 2 - 2x1 MUX ile Temel Lojik Kapıların Tasarımı

2.1 Teorik Araştırma

2:1 Mux yapısı ile AND, OR, NAND, NOR kapılarının nasıl oluşturulacağı hakkında teorik araştırma yapıldı.

A	B	OUT	
0	0	0	OUT = 0 when A = 0
0	1	0	
1	0	0	OUT = B when A = 1
1	1	1	

Şekil 5 AND kapısı Doğruluk Tablosu

Şekil 6 2:1 Mux kullanarak AND kapısının uygulanması

A	B	OUT	
0	0	1	OUT = 1 when A = 0
0	1	1	
1	0	1	OUT = B' when A = 1
1	1	0	

Şekil 7 NAND Kapısı Doğruluk Tablosu

Şekil 8 2:1 Mux kullanarak NAND kapısının uygulanması

A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	1

OUT = B when A = 0

OUT = 1 when A = 1

Şekil 9 OR Kapısı Doğruluk Tablosu

Şekil 10 2:1 Mux kullanarak OR kapısının uygulanması

A	B	OUT
0	0	1
0	1	0
1	0	0
1	1	0

OUT = B' when A = 0

OUT = 0 when A = 1

Şekil 11 NOR Kapısı Doğruluk Tablosu

Şekil 12 2:1 Mux kullanarak NOR kapısının uygulanması

2.2 DENEYİN YAPILIŞI

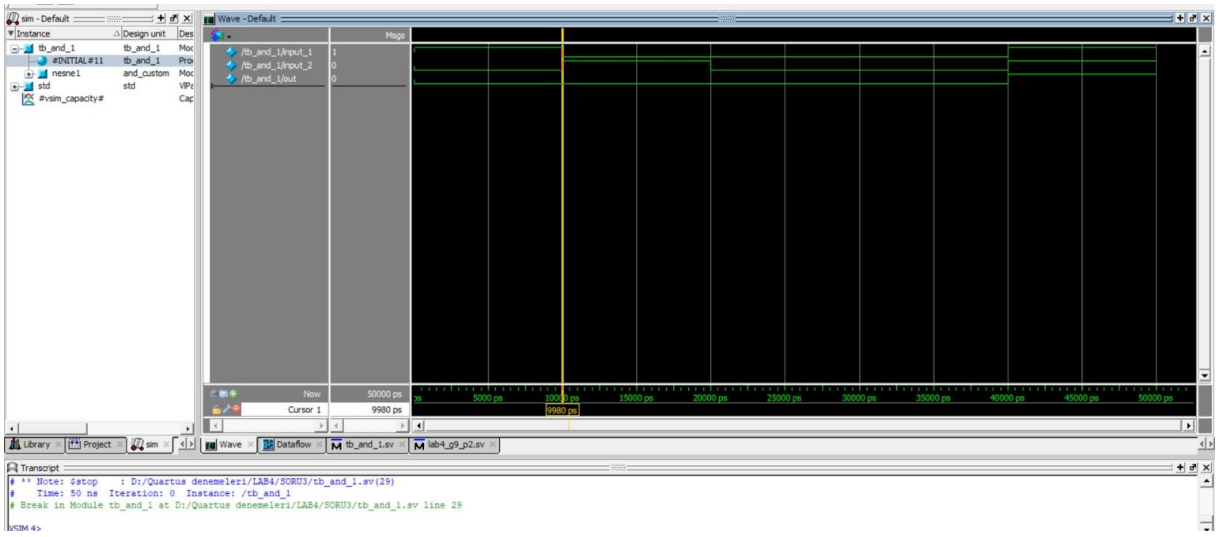
And kapısı:

```
module lab4_g9_p2(
    input_1, input_2,  s, out );
    input input_1;
    input input_2;
    input s;
    output out;
    assign out = (((s&input_1)) & ((s&input_2)));
endmodule
module and_custom(
      input logic in1,in2,
      output out1
);
lab4_g9_p2 nesne(.s(in1),.input_1(0),.input_2(in2),.out(out1));
endmodule
```

Tablo 3: Problem 2

```
`timescale 1ns/1ps
module tb_and_1 ();
logic  input_1, input_2;
reg  out;
and_custom nesne1(.in1(input_1), .in2(input_2), .out1(out));
initial begin
input_1=1; input_2=0;  #10;
input_1=0;  input_2=1;  #10;
input_2=0;  #10;  #10;
input_1=1; input_2=1;
#10;
 $stop;
 end
endmodule
```

Tablo 4: Problem 2 Testbench



Şekil 13 Problem 2 ModelSim Dalga Şematiği

OR kapısı:

```
//**OR MAIN*
/*lab4_g9_p2.sv
* Hazirlayanlar:
* Selen Erdogan - Ayse Serra Simsek
* Notlar: * ELM235 2020 Bahar
*
*/

module qu2_or(
     input logic in1,in2,
     output out1
);

qu2_or nesne(.s(in1),.input_1(in2),.input_2(1),.out(out1));

endmodule
```

Tablo 5: OR kapısı

```
//OR TESTBENCH
`timescale 1ns/1ps
module qu2_or_tb ();

logic  input_1, input_2;
reg  out;

qu2_or inst0(.in1(input_1), .in2(input_2), .out1(out));

initial begin

input_1=1; input_2=0;  #10;

input_1=0;  input_2=1;  #10;


input_2=0;  #10;  #10;

input_1=1; input_2=1;

#10;


 $stop;

 end

endmodule
```

Tablo 6: Or Kapısı Testbench

NAND Kapısı:

```
/*lab4_g9_p2.sv
* Hazırlayanlar:
* Selen Erdoğan - Ayşe Serra Şimşek
* Notlar: * ELM235 2020 Bahar
*
*/


module nand_1(
     input logic in1,in2,
     output out1
);

wire aracikis;
lab4_g9_p2 nesne1(.s(in2),.input_1(1),.input_2(0),.out(aracikis));
lab4_g9_p2
nesne2(.s(in1),.input_1(1),.input_2(aracikis),.out(out1));
endmodule
```

Tablo 7: NAND Kapısı

```
`timescale 1ns/1ps
module tb_nand_1 ();
logic  input_1, input_2;
reg  out;
nand_1 inst0(.in1(input_1), .in2(input_2), .out1(out));

initial begin
input_1=1; input_2=0;  #10;
input_1=0;  input_2=1;  #10;
input_2=0;  #10;  #10;
input_1=1; input_2=1;
#10;

 $stop;

 end

endmodule
```

Tablo 8: NAND Kapısı Testbench

NOR Kapısı:

```
/*lab4_g9_p2.sv
* Hazırlayanlar:
* Selen Erdoğan - Ayşe Serra Şimşek
* Notlar: * ELM235 2020 Bahar
*
*/

module nor_1(
     input logic in1,in2,
     output out1


);

wire aracikis;

lab4_g9_p2 nesne1(.s(in2),.input_1(1),.input_2(0),.out(aracikis));
lab4_g9_p2
nesne2(.s(in1),.input_1(aracikis),.input_2(0),.out(out1));

endmodule
```

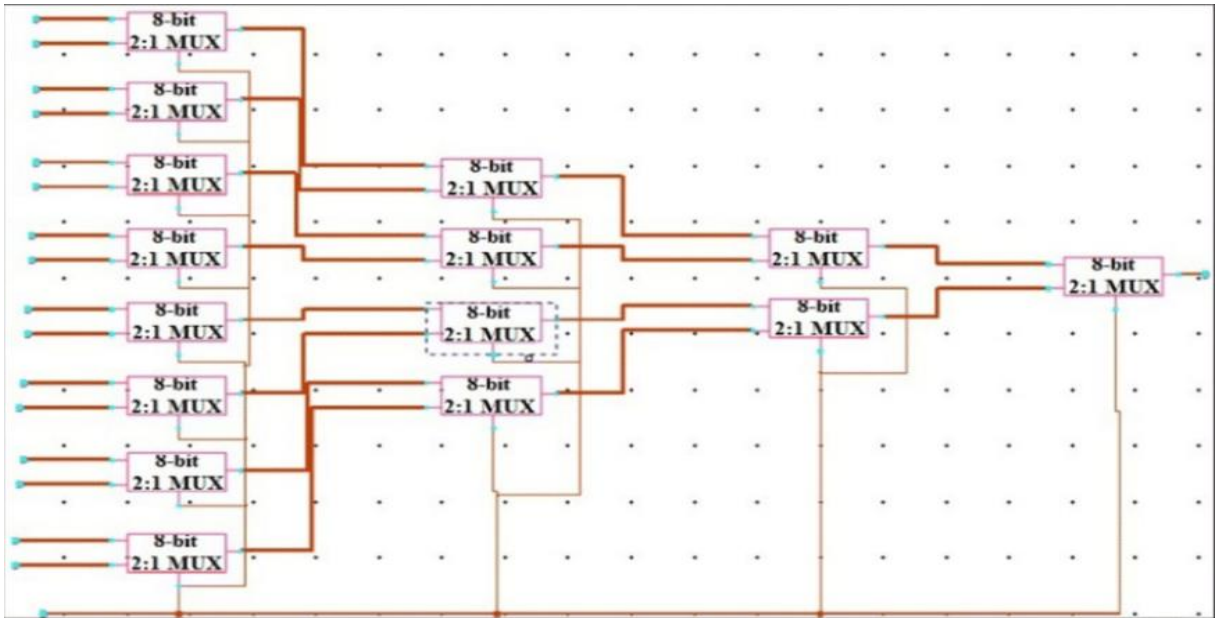
Tablo 9: NOR kapısı

```
`timescale 1ns/1ps
module tb_nor_1 ();
logic  input_1, input_2;
reg  out;

nor_1 inst0(.in1(input_1), .in2(input_2), .out1(out));

initial begin

input_1=1; input_2=0;  #10;
input_1=0;  input_2=1;  #10;

input_2=0;  #10;  #10;
input_1=1; input_2=1;
#10;

 $stop;
 end
endmodule
```

Tablo 10: NOR kapısı Testbench

Problem 3 - 16x1 MUX Tasarımı

Teorik Araştırma:



Şekil 14 2:1 Mux yapısı kullanarak 16:1 Mux yapısı oluşturma

S_3	S_2	S_1	S_0	Y
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8
1	0	0	1	I_9
1	0	1	0	I_{10}
1	0	1	1	I_{11}
1	1	0	0	I_{12}
1	1	0	1	I_{13}
1	1	1	0	I_{14}
1	1	1	1	I_{15}

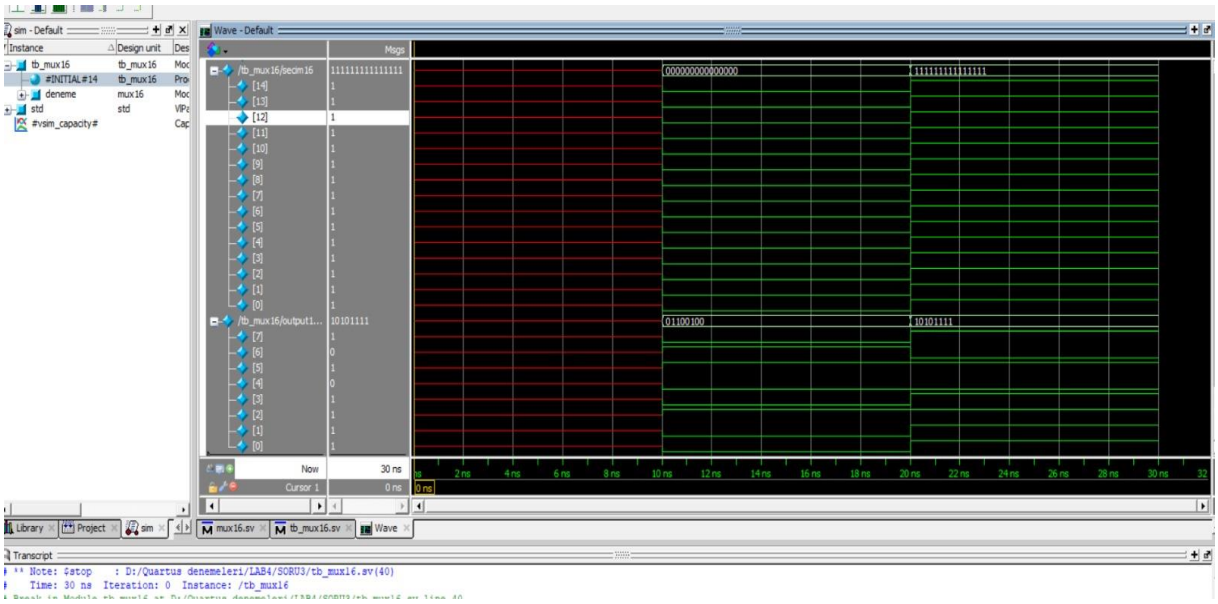
Şekil 15 16:1 Mux Doğruluk Tablosu

```
module mux16(
input logic [7:0] a1,a2,a3,a4,b1,b2,b3,b4,c1,c2,c3,c4,d1,d2,d3,d4,
input logic [14:0] secim16,
output logic [7:0] output16lik
);
wire [7:0] aracikis1,aracikis2;
mux8 ilkmux(a1,a2,a3,a4,b1,b2,b3,b4,secim16[14:8],aracikis1);
mux8 ikincimux(c1,c2,c3,c4,d1,d2,d3,d4,secim16[7:1],aracikis2);
mux2 ucuncumux(aracikis1,aracikis2,secim16[0],output16lik);

endmodule

module mux8(
input logic [7:0] a1,a2,b1,b2,c1,c2,d1,d2,
input logic [6:0] secim8,
output logic [7:0] output8lik
);

wire [7:0] aracikis1,aracikis2;
mux4 ilkmux         (a1,a2,b1,b2,secim8[6:4],aracikis1);
mux4 ikincimux      (c1,c2,d1,d2,secim8[3:1],aracikis2);
mux2  ucuncumux (aracikis1,aracikis2,secim8[0],output8lik);

endmodule

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

endmodule


module mux2 (
input logic[7:0]   d0,d1,
input logic        s,
output logic[7:0]  y       );

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

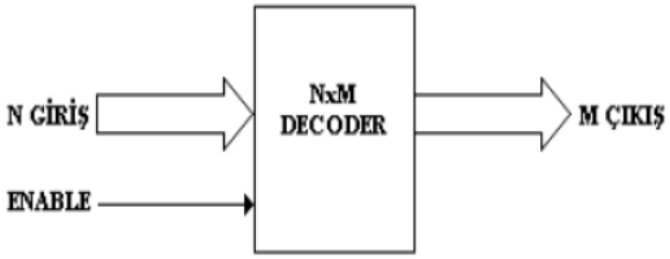
Tablo 11: Problem 3

```
`timescale 1ns/1ns

module tb_mux16();

 logic [7:0] a1,a2,a3,a4,b1,b2,b3,b4,c1,c2,c3,c4,d1,d2,d3,d4;

logic [14:0]  secim16;

logic [7:0] output16lik;


mux16 deneme(
.a1(a1),.a2(a2),.a3(a3),.a4(a4),.b1(b1),.b2(b2),.b3(b3),.b4(b4),.c1(
c1),.c2(c2),.c3(c3),.c4(c4),.d1(d1),.d2(d2),.d3(d3),.d4(d4),.secim16
(secim16),.output16lik(output16lik) );

 initial begin
a1=100;
a2=200;
a3=250;
a4=90;
b1=50;
b2=25;
b3=30;
b4=60;
c1=10;
c2=20;
c3=40;
c4=35;
d1=75;
d2=150;
d3=80;
d4=175;
#10;

 secim16= 15'b0000_0000_0000_000;
#10;
  secim16= 15'b1111_1111_1111_111;
#10;



 $stop;

 end
endmodule
```

Tablo 12: Problem 3 Testbench



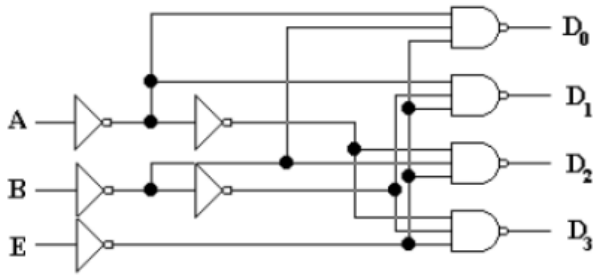
Şekil 16 Problem 3 ModelSim Dalga Şematiği



Şekil.1 NxM Decoder Genel Görünüşü

Problem 4 - 2x4 Kod Çözücü tasarımı

Kod Çözücü (Decoder): Decoder, N giriş hattından oluşan binary giriş bilgisini 2 N çıkış hattına çevirebilen kombinasyonel devredir. Decoderler şekilde görüldüğü gibi yapı olarak N binary giriş hattını M çıkış hattına çevirdiklerinden dolayı NxM ya da N-M decoder olarak adlandırılır. Burada M=2 N ilişkisi söz konusudur.



GİRİŞLER			ÇIKIŞLAR			
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Şekil.2 2x4 Decoder Devresi ve Doğruluk Tablosu

```
module top(
input logic [7:0] a1,a2,a3,a4,b1,b2,b3,b4,c1,c2,c3,c4,d1,d2,d3,d4,
input logic [3:0] secim16,
output logic [7:0] out
);
wire [15:0] secim;
decoder dekod(secim16,secim);
mux16
my_mux(.a1(a1),.a2(a2),.a3(a3),.a4(a4),.b1(b1),.b2(b2),.b3(b3),.b4(b
4),.c1(c1),.c2(c2),.c3(c3),.c4(c4),.d1(d1),.d2(d2),.d3(d3),.d4(d4),.
secim(secim),.out(out));


endmodule

module mux16(
input logic [7:0] a1,a2,a3,a4,b1,b2,b3,b4,c1,c2,c3,c4,d1,d2,d3,d4,
input logic [14:0] secim,
output logic [7:0] out
);

wire [7:0] tercih1,tercih2;
mux8 ilkmux(a1,a2,a3,a4,b1,b2,b3,b4,secim[14:8],tercih1);
mux8 ikincimux(c1,c2,c3,c4,d1,d2,d3,d4,secim[7:1],tercih2);
mux2 ucuncumux(tercih1,tercih2,secim[0],out);


endmodule

module mux8(
input logic [7:0] a1,a2,b1,b2,c1,c2,d1,d2,
input logic [6:0] secim1,
output logic [7:0] out
);

wire [7:0] tercih1,tercih2;

mux4 ilkmux(a1,a2,b1,b2,secim[6:4],tercih1);
mux4 ikincimux(c1,c2,d1,d2,secim[3:1],tercih2);
mux2 ucuncumux(tercih1,tercih2,secim[0],out);

endmodule


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

```
wire [7:0] tercih1,tercih2 ;

mux2 ilkmux(a,b,secim[2],tercih1);
mux2 ikincimux(c,d,secim[1],tercih2);
mux2 ucuncumux(tercih1,tercih2,secim[0],out);

endmodule

module mux2(
input logic [7:0] giris1,giris2,
input logic secim,
output logic [7:0] cikis
);

assign cikis = secim ? giris1 : giris2 ;
endmodule


module decoder (
   input logic [3:0]   giris,
   output logic[15:0] cikis
);

parameter deger = 16'b0000_0000_0000_0001;
assign cikis = (giris == 4'b0000) ? deger   :
               (giris == 4'b0001) ? deger<<1:
               (giris == 4'b0010) ? deger<<2:
               (giris == 4'b0011) ? deger<<3:
               (giris == 4'b0100) ? deger<<4:
               (giris == 4'b0101) ? deger<<5:
               (giris == 4'b0110) ? deger<<6

               (giris == 4'b0111) ? deger<<7:
               (giris == 4'b1000) ? deger<<8:
               (giris == 4'b1001) ? deger<<9:
               (giris == 4'b1010) ? deger<<10:
               (giris == 4'b1011) ? deger<<11:
               (giris == 4'b1100) ? deger<<12:
               (giris == 4'b1101) ? deger<<13:
               (giris == 4'b1110) ? deger<<14:
               (giris == 4'b1111) ? deger<<15:
16'bxxxx_xxxx_xxxx_xxxx;

endmodule
```

Tablo 13: Problem 4

```
`timescale 1ns / 1ps
module tb_mux16();
logic [7:0] a1,a2,a3,a4,b1,b2,b3,b4,c1,c2,c3,c4,d1,d2,d3,d4;
logic [3:0] secim16;
logic [7:0] out;
top
deneme(.a1(a1),.a2(a2),.a3(a3),.a4(a4),.b1(b1),.b2(b2),.b3(b3),.b4(b
4),.c1(c1),.c2(c2),.c3(c3),.c4(c4),.d1(d1),.d2(d2),.d3(d3),.d4(d4),.
secim16(secim16),.out(out));
initial begin
a1=10;
a2=20;
a3=30;
a4=40;
b1=50;
b2=60;
b3=70;
b4=80;
c1=90;
c2=100;
c3=110;
c4=120;
d1=130;
d2=140;
d3=150;
d4=160;
#5;
secim16= 4'b0001  ;
#5;
secim16= 4'b1111   ;
#5;
secim16= 4'b1100   ;
#5;
secim16=4'b0010 ;
#5;
secim16=4'b0000;
end
endmodule
```

Tablo 14: Problem 4 Testbench

3. Sonuçlar ve Genel Yorumlar

Bu laboratuvar çalışmasında multiplexer yapısı ve şematik ortamda oluşturulması ile ilgili bilgiler edinildi. Bir çok giriş hattından gelen bilgilerden birisini seçerek uygun çıkış hattına yönlendirilmesini sağlayan birleşik devrelere “çoklayıcı / veri seçici devreler” (multiplexer) denildiği öğrenildi. Daha sonra edinilen bu yapı diğer problem çözümlerinde kullanıldı. 2x1 MUX devresi şematik düzeyde sadece NAND kapıları kullanarak tasarlandı. Testbench

oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemlendi. ModelSim dalga şeması elde edildi. Temel lojik kapıların 2:1 Mux yapısı ile nasıl elde edildiği araştırıldı. Elektronik ortamda simüle edilen bu devreler föye konuldu. Dalga şematikleri incelendi. ModelSim kullanılırken bazı kodlarda "Error loading design" hatası alındı. Gerekli araştırmalar yapıldığı halde dosya yolunda veyahut dosya isimlerinde hata bulunamadı.

4. Referanslar

https://www.tutorialspoint.com/digital_circuits/digital_circuits_multiplexers.htm

https://320volt.com/coklayicilar-veri-seciciler-multiplexers-data-selector/

https://vlsiuniverse.blogspot.com/2016/08/logic-functions-using-mux.html

http://dosya.kmu.edu.tr/eemuh/userfiles/files/Dersler/Say%C4%B1sal%20Elektronik-2/Say%C4%B1sal%20Elektronik-II%20Deney2.pdf

https://technobyte.org/verilog-multiplexer-2x1/

https://technobyte.org/verilog-multiplexer-8x1/