GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

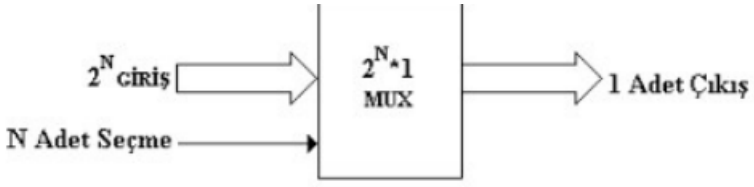
LAB 4 Deney Raporu

Kombinasyonel Devreler Tabanlı Tasarım.

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

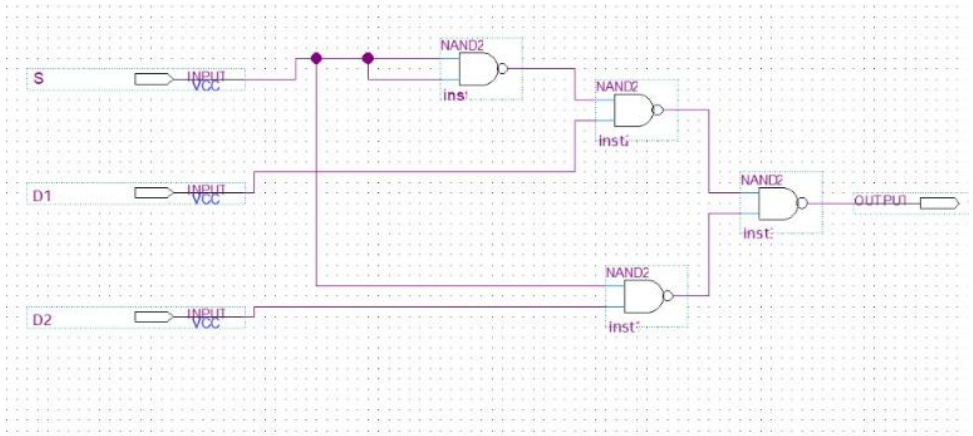
PROBLEMLER

Problem 1 – 2x1 MUX Tasarımı

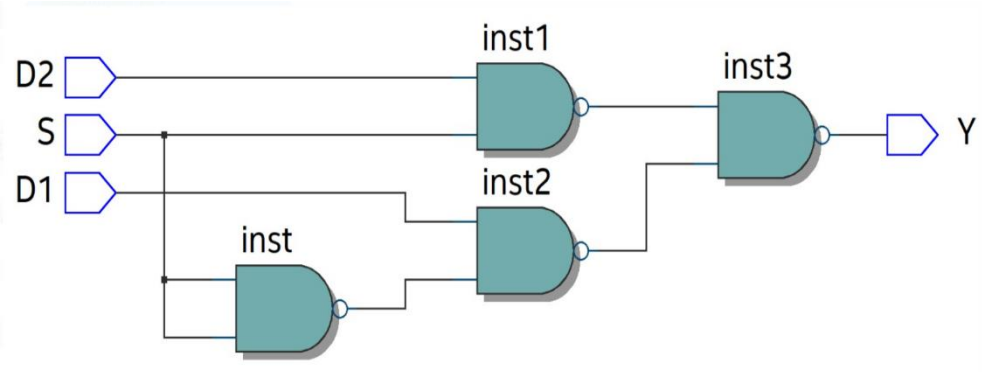


Şekil 1 2ⁿx1 Çoğullacı

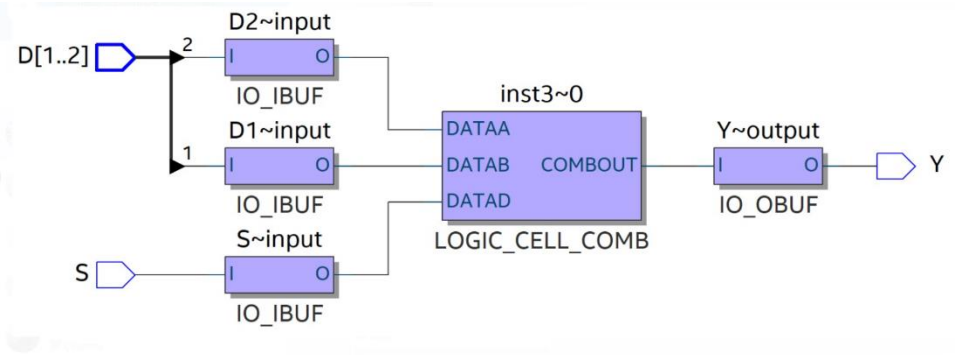
A. 2x1 MUX devresini şematik düzeyde sadece NAND kapıları kullanarak tasarlayınız.
B. Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemleyin
. C. ModelSim dalga şemasını File→Export→Image.. ile PNG olarak çıkarabilirsiniz. Veya tüm ModelSim projesinin ekran görüntüsü alabilirsiniz.



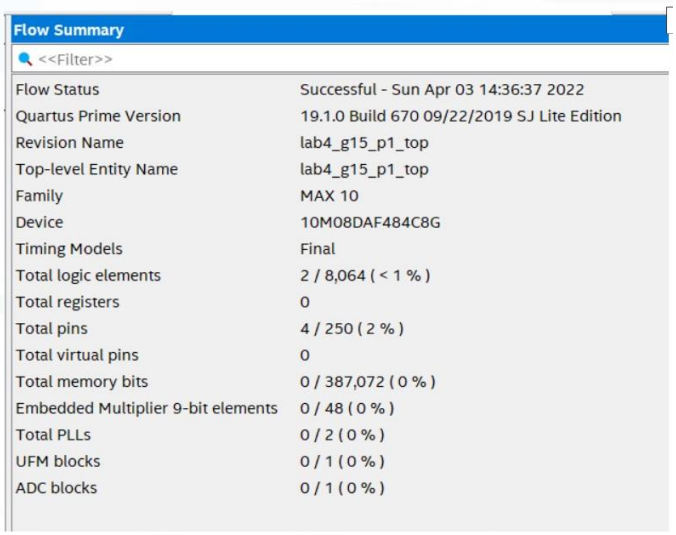
Şekil 1: Problem 1'in Quartus Prime'da şematik çizimi



Şekil 2: Problem 1 'in RTL şeması



Şekil 3: Problem 1'in Post Mapping şeması



Flow Summary

<<Filter>>

Flow Status	Successful - Sun Apr 03 14:36:37 2022
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	lab4_g15_p1_top
Top-level Entity Name	lab4_g15_p1_top
Family	MAX 10
Device	10M08DAF484C8G
Timing Models	Final
Total logic elements	2 / 8,064 (< 1 %)
Total registers	0
Total pins	4 / 250 (2 %)
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 4: Problem1'in akış özeti

SORU 1 TESTBENCH:

```
`timescale 1ns/1ps
module lab4_tb_a ();

logic D1,D2,S;
logic Y;

lab4_p2_g15 uut0(.D1(D1), .D2(D2) ,.S(S), .Y(Y));

initial begin
D1=0; D2=0; S=0; #5;
D1=0; D2=0; S=1; #5;
D1=0; D2=1; S=0; #5;
D1=0; D2=1; S=1; #5;
D1=1; D2=0; S=0; #5;
D1=1; D2=0; S=1; #5;
D1=1; D2=1; S=0; #5;
D1=1; D2=1; S=1; #5;

$stop;
end
endmodule
```

C:/quartus_introo/tb_lab4_g15_p1.sv (/tb_lab4_g15_p1) - Default

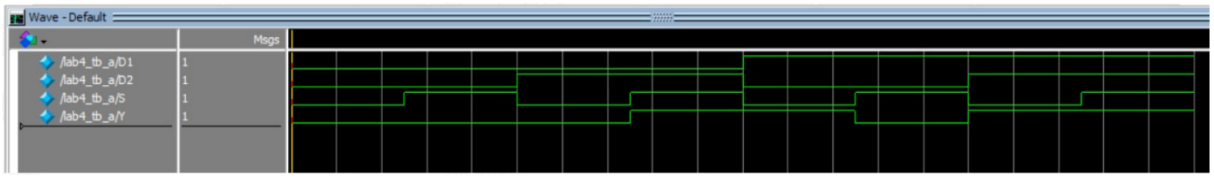
```
Ln#
1   `timescale 1ns/1ps
2   module tb_lab4_g15_p1();
3
4   logic A,B,S;
5   logic pin_name1;
6
7   tb_lab4_g15_p1 uut0(.A(A), .B(B) ,.S(S), .pin_name1(pin_name1));
8
9   initial begin
10  A=0; B=0; S=0; #5;
11  A=0; B=0; S=1; #5;
12  A=0; B=1; S=0; #5;
13  A=0; B=1; S=1; #5;
14  A=1; B=0; S=0; #5;
15  A=1; B=0; S=1; #5;
16  A=1; B=1; S=0; #5;
17  A=1; B=1; S=1; #5;
18
19  $stop;
20  end
21  endmodule
22
23
```

Şekil 5:

SORU 1 .v Dosyası:

```
module lab4_p2_g15(
      S,
      D1,
      D2,
      Y
);


input wire S;
input wire D1;
input wire D2;
output wire      Y;

wire  SYNTHESIZED_WIRE_0;
wire  SYNTHESIZED_WIRE_1;
wire  SYNTHESIZED_WIRE_2;




assign      SYNTHESIZED_WIRE_0 = ~(S & S);

assign      SYNTHESIZED_WIRE_1 = ~(D2 & S);

assign      SYNTHESIZED_WIRE_2 = ~(D1 & SYNTHESIZED_WIRE_0);

assign      Y = ~(SYNTHESIZED_WIRE_1 & SYNTHESIZED_WIRE_2);


endmodule
```

WAVE

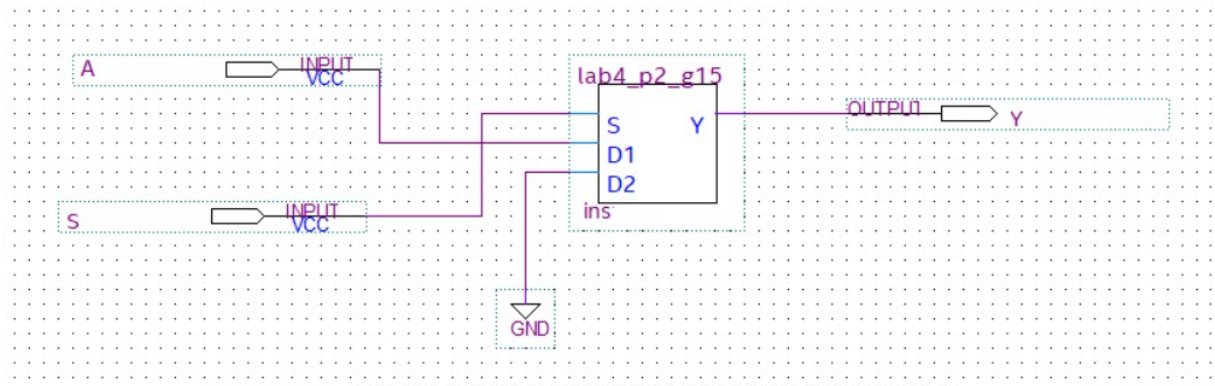


Şekil 6: Problem 1 için dalga şekli

Problem 2 – 2x1 MUX ile Temel Lojik Kapıların Tasarımı

A. Sadece Problem 1'de tasarladığınız 2x1 MUX'u kullanarak AND, OR, NAND, ve NOR kapılarını tasarlayınız.
B. Testbench oluşturarak, devrelerin bütün girişlere karşı nasıl davrandığını gözlemleyin.
C. Çıkan dalga şekillerini yorumlayın.

A)



Şekil 7: roblem 1'de tasarladığImız 2x1 MUX'u kullanarak AND kapısının tasarlanması

SORU 2 AND İÇİN TESTBENCH

```
`timescale 1ns/1ps
module tb_kapilar ();

logic A;
logic S;
logic Y; // all the outputs

aykapilarson uut0(.A(A), .S(S), .Y(Y));
initial begin
A = 0;S=0; #10;
A = 0;S=1; #10;
A = 1;S=0; #10;
A = 1;S=1; #10;
#20; // wait 20 ns after completion
$stop; // stop the simulation
end
endmodule
```

C:/p4lab4/tb_kapilar.sv (/tb_kapilar) - Default

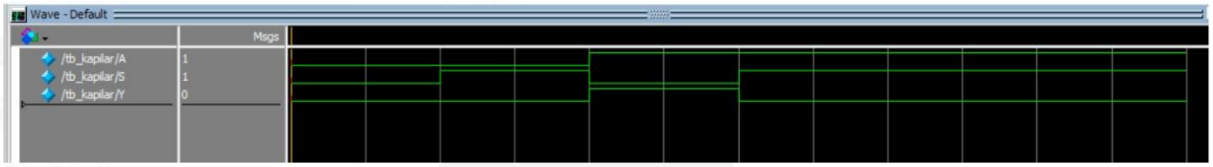
```
Ln#
 1   `timescale 1ns/1ps
 2   module tb_kapilar ();
 3
 4   logic A;
 5   logic S;
 6   logic Y; // all the outputs
 7
 8   aykapilarson uut0(.A(A), .S(S), .Y(Y));
 9   initial begin
10   A = 0;S=0; #10;
11   A = 0;S=1; #10;
12   A = 1;S=0; #10;
13   A = 1;S=1; #10;
14   #20; // wait 20 ns after completion
15   $stop; // stop the simulation
16   end
17   endmodule
18
```

Şekil 8: SORU2 soru2 and kapısı için testbench

SORU 2 AND İÇİN .V KODLARI

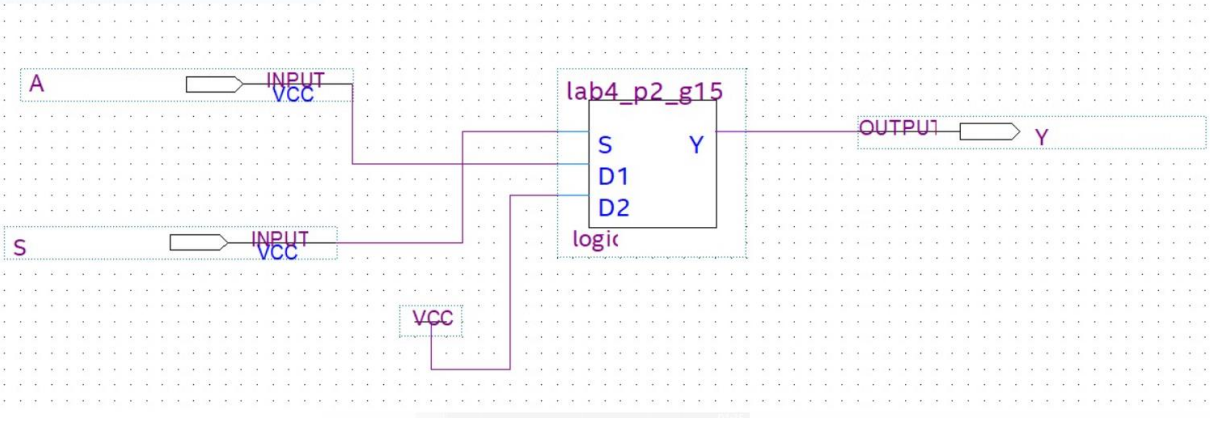
```
module aykapilarson(
        A,
        S,
        Y
);


input wire      A;
input wire      S;
output wire     Y;

wire    SYNTHESIZED_WIRE_0;

assign  SYNTHESIZED_WIRE_0 = 0;



lab4_p2_g15    b2v_inst(
        .S(S),
        .D1(A),
        .D2(SYNTHESIZED_WIRE_0),
        .Y(Y));



endmodule
```

WAVE SORU 2 AND



Şekil 9: Soru2 AND kapısı için dalga şeması

SORU 2 OR KAPISI



Şekil 10: Soru 2 OR kapısı için Quartus Prime çizimi

SORU 2 OR İÇİN TESTBENCH

```
`timescale 1ns/1ps
module tb_kapilar ();

logic A;
logic S;
logic Y; // all the outputs

aykapilarson uut0(.A(A), .S(S), .Y(Y));
initial begin
A = 0;S=0; #10;
A = 0;S=1; #10;
A = 1;S=0; #10;
A = 1;S=1; #10;
#20; // wait 20 ns after completion
$stop; // stop the simulation
end
endmodule
```

C:/p4lab4/tb_kapilar.sv (/tb_kapilar) - Default

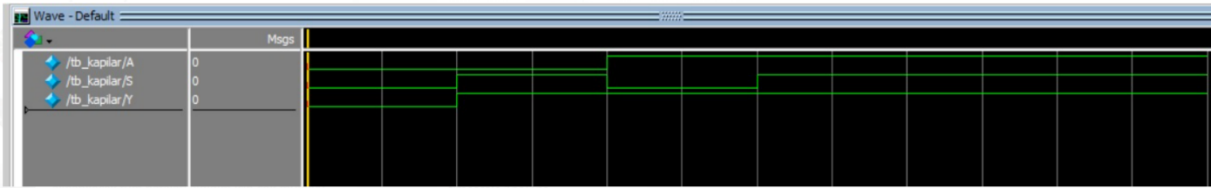
```
Ln#
 1  `timescale 1ns/1ps
 2  module tb_kapilar ();
 3
 4  logic A;
 5  logic S;
 6  logic Y; // all the outputs
 7
 8  aykapilarson uut0(.A(A), .S(S), .Y(Y));
 9  initial begin
10  A = 0;S=0; #10;
11  A = 0;S=1; #10;
12  A = 1;S=0; #10;
13  A = 1;S=1; #10;
14  #20; // wait 20 ns after completion
15  $stop; // stop the simulation
16  end
17  endmodule
18
```

Şekil 11: SORU2 soru2 OR kapısı için testbench

SORU 2 OR İÇİN .V KODLAR

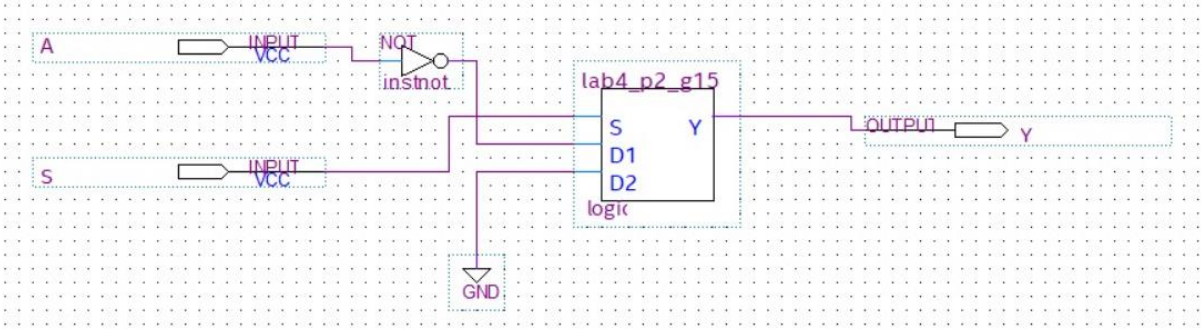
```
module aykapilarson(
      A,
      S,
      Y
);


input wire A;
input wire S;
output wire      Y;

wire  SYNTHESIZED_WIRE_0;

assign     SYNTHESIZED_WIRE_0 = 1;



lab4_p2_g15      b2v_logic(
      .S(S),
      .D1(A),
      .D2(SYNTHESIZED_WIRE_0),
      .Y(Y));


endmodule
```

WAVE SIRU 2 OR



Şekil 12: Soru2 OR kapısı için dalga şeması

SORU 2 NOR KAPISI



Şekil 13: Soru 2 NOR kapısı için Quartus Prime çizimi

SORU 2 OR İÇİN TESTBENCH

```
`timescale 1ns/1ps
module tb_kapilar ();

logic A;
logic S;
logic Y; // all the outputs

aykapilarson uut0(.A(A), .S(S), .Y(Y));
initial begin
A = 0;S=0; #10;
A = 0;S=1; #10;
A = 1;S=0; #10;
A = 1;S=1; #10;
#20; // wait 20 ns aft er completion
$stop; // stop the simulation
end
endmodule
```

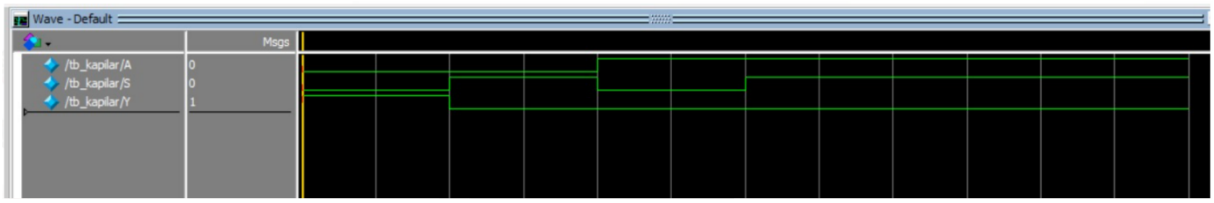
```
1   `timescale 1ns/1ps
2   module tb_kapilar ();
3
4   logic A;
5   logic S;
6   logic Y; // all the outputs
7
8   aykapilarson uut0(.A(A), .S(S), .Y(Y));
9   initial begin
10  A = 0;S=0; #10;
11  A = 0;S=1; #10;
12  A = 1;S=0; #10;
13  A = 1;S=1; #10;
14  #20; // wait 20 ns aft er completion
15  $stop; // stop the simulation
16  end
17  endmodule
18
```

Şekil 14: SORU2 soru2 NOR kapısı için testbench

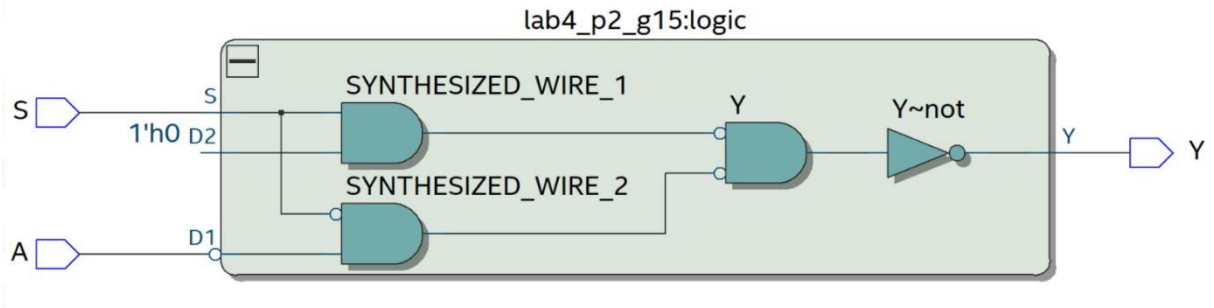
SORU 2 NOR İÇİN .V KODLAR

```
module aykapilarson(
      A,
      S,
      Y
);


input wire A;
input wire S;
output wire      Y;

wire  SYNTHESIZED_WIRE_0;
wire  SYNTHESIZED_WIRE_1;

assign      SYNTHESIZED_WIRE_1 = 0;




assign      SYNTHESIZED_WIRE_0 =  ~A;


lab4_p2_g15      b2v_logic(
      .S(S),
      .D1(SYNTHESIZED_WIRE_0),
      .D2(SYNTHESIZED_WIRE_1),
      .Y(Y));


endmodule
```

WAVE SORU 2 NOR

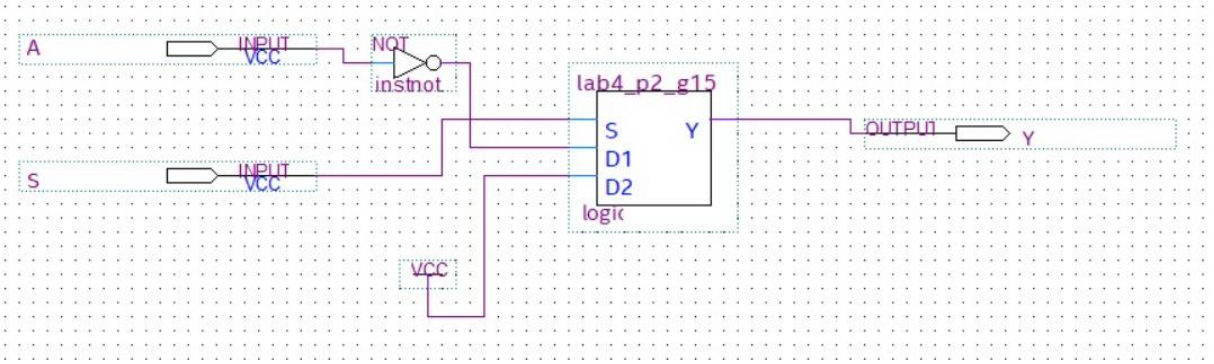


Şekil 15: Soru2 NOR kapısı için dalga şeması



Şekil 16: soru 2 NOR kapısı için RTL şeması

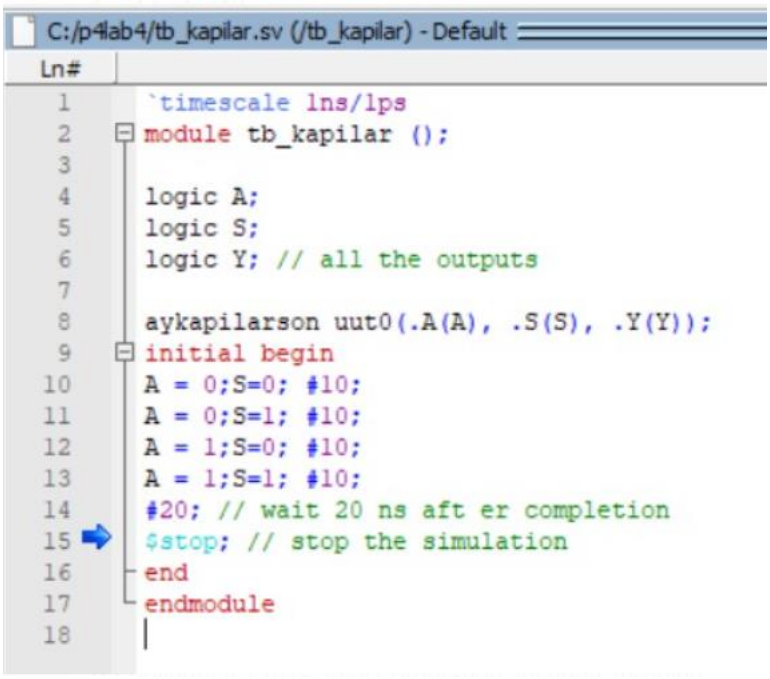
SORU 2 NAND KAPISI



Şekil 17: Soru 2 NAND kapısı için Quartus Prime çizimi

SORU 2 NAND İÇİN TESTBENCH

```
`timescale 1ns/1ps
module tb_kapilar ();

logic A;
logic S;
logic Y; // all the outputs

aykapilarson uut0(.A(A), .S(S), .Y(Y));
initial begin
A = 0;S=0; #10;
A = 0;S=1; #10;
A = 1;S=0; #10;
A = 1;S=1; #10;
#20; // wait 20 ns aft er completion
$stop; // stop the simulation
end
endmodule
```

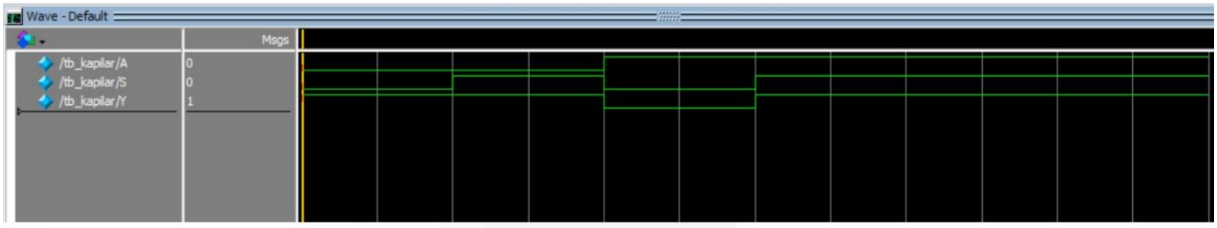


Şekil 18: SORU 2 NAND kapısı için testbench

SORU 2 NAND İÇİN KODLAR

```
module aykapilarson(
      A,
      S,
      Y
);


input wire A;
input wire S;
output wire      Y;

wire  SYNTHESIZED_WIRE_0;
wire  SYNTHESIZED_WIRE_1;

assign      SYNTHESIZED_WIRE_1 = 1;



assign      SYNTHESIZED_WIRE_0 =  ~A;


lab4_p2_g15      b2v_logic(
      .S(S),
      .D1(SYNTHESIZED_WIRE_0),
      .D2(SYNTHESIZED_WIRE_1),
      .Y(Y));


endmodule
```

WAVE SORU2 NAND İÇİN



Şekil 19: Soru2 NAND kapısı için dalga şeması

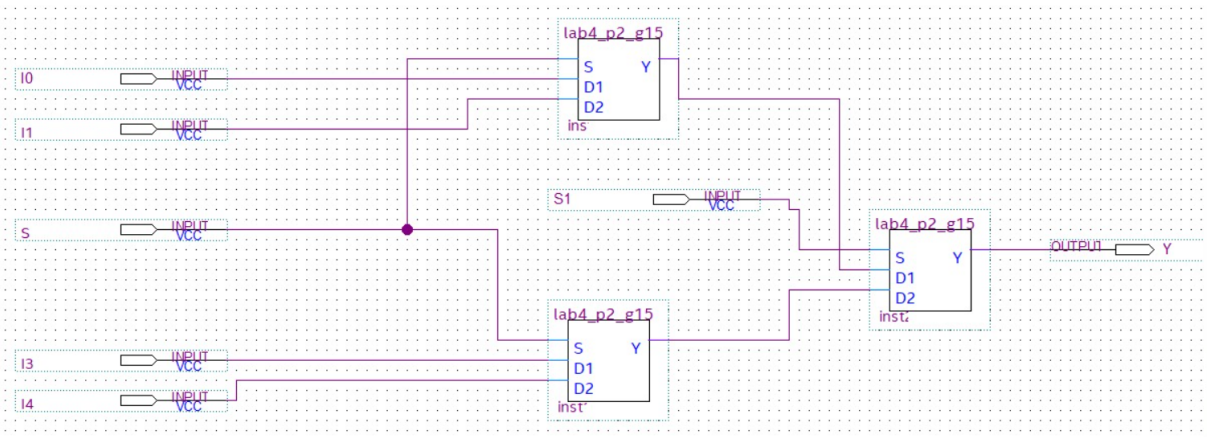
SORU 3

Problem 3 – 16x1 MUX Tasarımı A. Problem 1' deki 2x1 MUX'u kullanarak hiyerarşik olarak sırasıyla 4x1, 8x1 ve 16x1 MUX devrelerini tasarlayınız. 4x1 MUX'u 8x1 MUX'da ve 8x1 MUX'u 16x1 MUX'da kullanabilirsiniz.

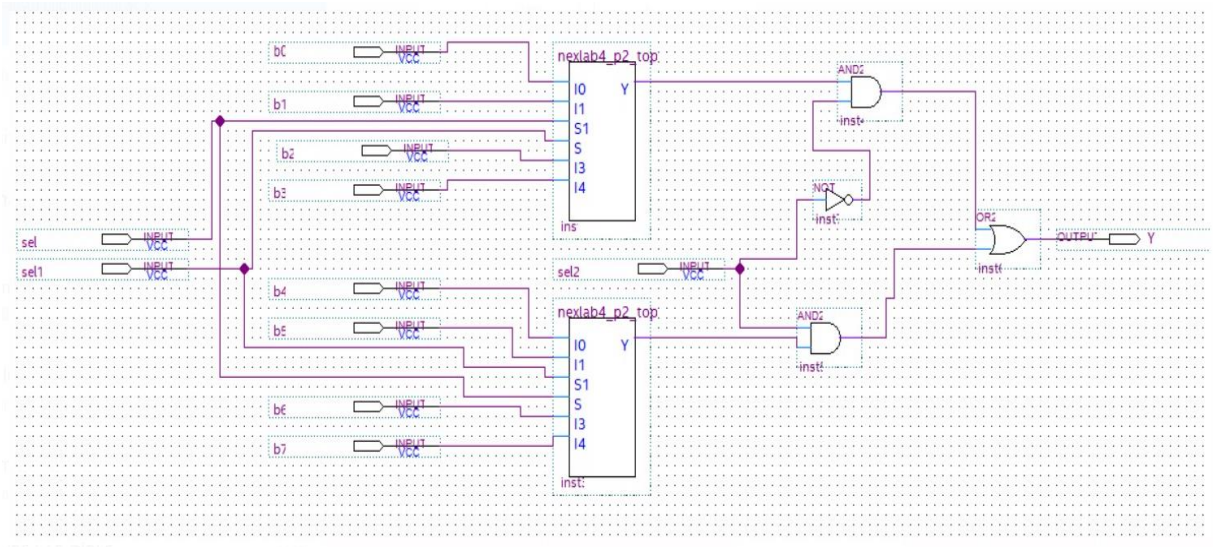
B. 16x1 MUX için Testbench oluşturarak, devrenin rastgele seçilmiş 4 giriş için farklı seçim girişine karşı nasıl davrandığını gözlemleyin.

TESTBENCH

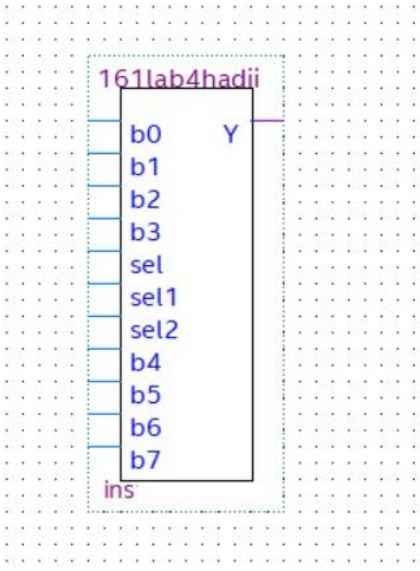
```
`timescale 1ns / 1ps
module test_lab4_p33();

logic [7:0] d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,d12,d13,d14,d15,d16;

logic sselect, select1, select2, select3;

logic [7:0] Y;
olsunhadi
uut0(.d1(d1),.d2(d2),.d3(d3),.d4(d4),.d5(d5),.d6(d6),.d7(d7),.d8(d8),.d9(d9),.d10(d10),.d11(d11),.
d12(d12),.d13(d13),.d14(d14),.d15(d15),.d16(d16),.sselect(sselect),.select1(select1),.select2(select
2),.select3(select3),.Y(Y));

initial begin

d1=10;
d2=20;
d3=30;
d4=40;
d5=50;
d6=60;
d7=70;
d8=80;
d9=90;
d10=100;
d11=110;
d12=120;
d13=130;
d14=140;
d15=150;
d16=160;

#5;
sselect= 4'b0001  ;
#5;
select1= 4'b1111   ;
#5;
select2= 4'b1100   ;
#5;
select3=4'b0010 ;
#5;
end

endmodule
```



Şekil 20: SORU 1'deki devreden 2X1 MUX

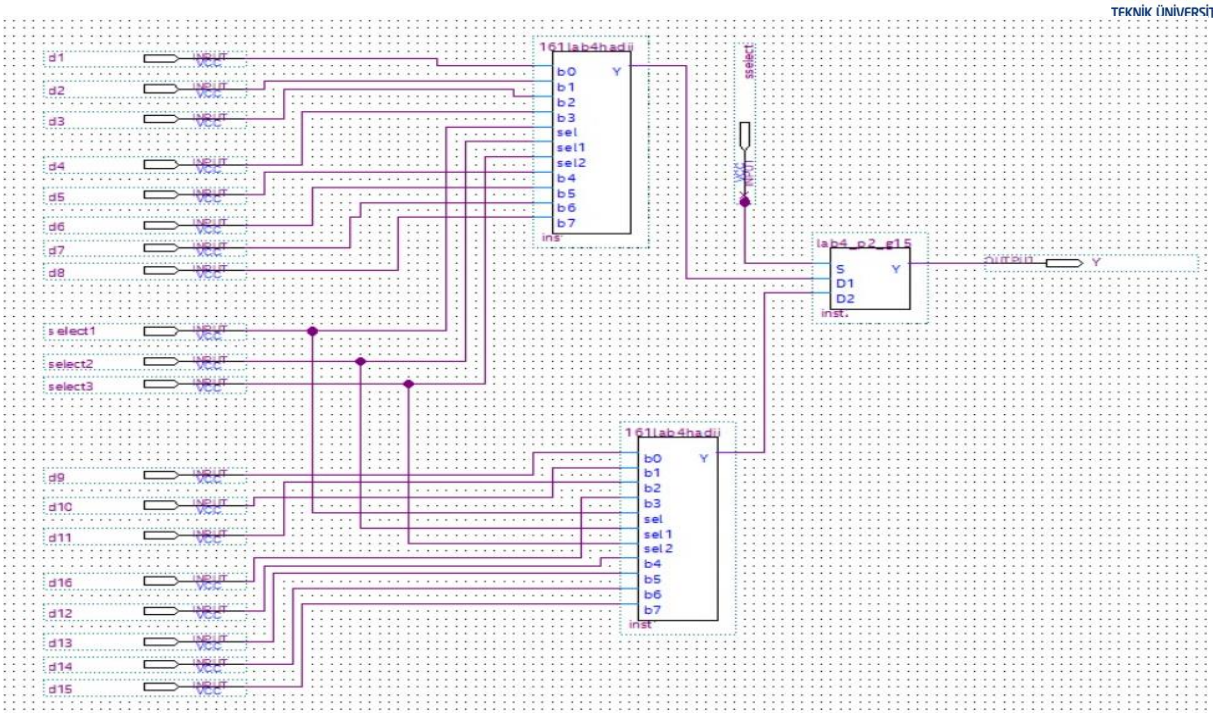


Şekil 21: 2X1 MUX 'dan 4X1 MUX eldesi

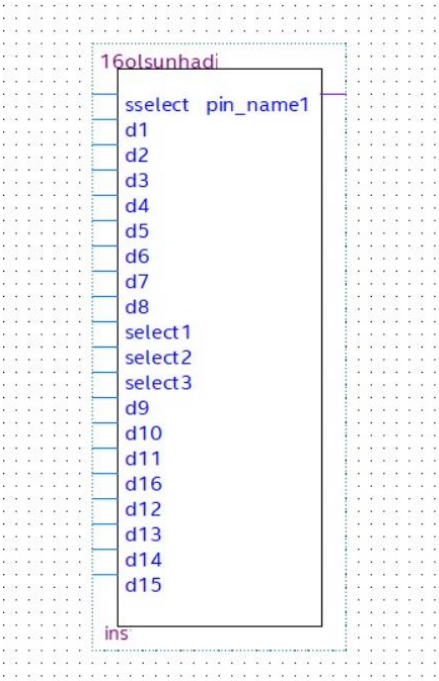


:

→ 4X1 MUX



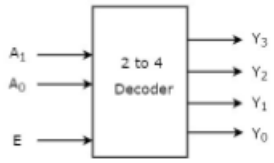
Şekil 22: 4X1 MUX'dan 8X1 MUX eldesi



16X1 MUX

SORU 4:

Problem 4 –2x4 Kod Çözücü tasarımı

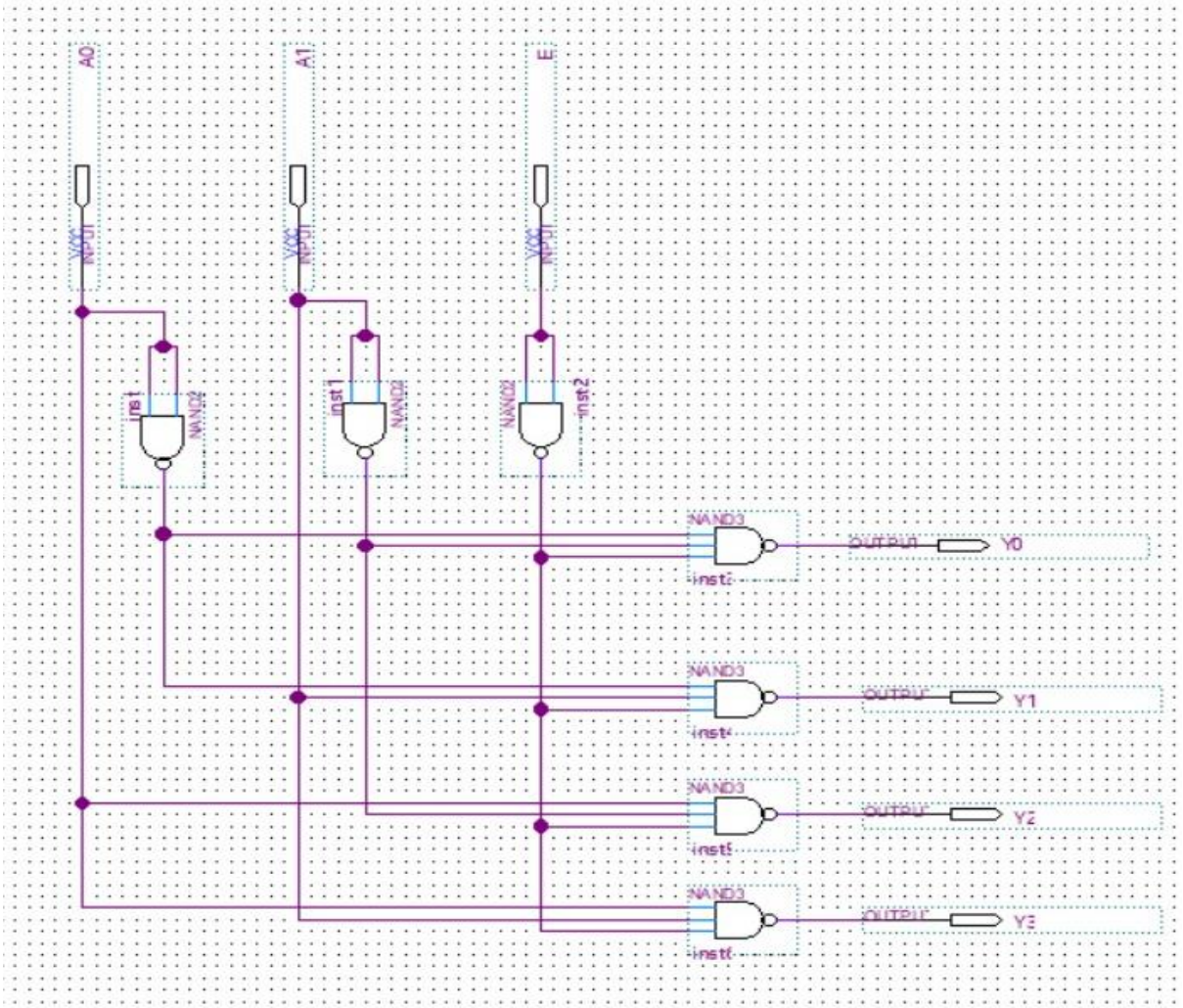


Sekil 2. 2X4 Kod çözücü

A. 2x4 Kod çözücü devresini Sadece NAND kapıları kullanarak tasarlayınız.

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



TESTBENCH

```
`timescale 1ns/1ps
module tb_lab4_g15_p4 ();

reg A;
reg B;
reg S;// all the inputs
reg Y0,Y1,Y2,Y3; // all the outputs

example_top uut0(.A(A), .B(B), .S(S),
.Y0(Y0),.Y1(Y1),.Y2(Y2),.Y3(Y3));
initial begin
A = 0; B =0; S=1; #10;
A = 0; B =0 ;S=0 ;#10;
A = 0; B =1 ;S=0; #10;
A = 1; B = 0 ;S=0; #10;
A = 1; B =1 ;S=0 ;#10;
#20; // wait 20 ns after completion
$stop; // stop the simulation
end
endmodule
```

```
module example_top(
      A,
      B,
      S,
      Y0,
      Y1,
      Y2,
      Y3
);


input wire A;
input wire B;
input wire S;
output wire      Y0;
output wire      Y1;
output wire      Y2;
output wire      Y3;

wire  SYNTHESIZED_WIRE_8;
wire  SYNTHESIZED_WIRE_9;
wire  SYNTHESIZED_WIRE_10;
```

.V KODLAR

```
assign     SYNTHESIZED_WIRE_8 = ~(A & A);

assign     SYNTHESIZED_WIRE_9 = ~(B & B);

assign     SYNTHESIZED_WIRE_10 = ~(S & S);

assign     Y0 = ~(SYNTHESIZED_WIRE_8 & SYNTHESIZED_WIRE_9 &
SYNTHESIZED_WIRE_10);

assign     Y1 = ~(SYNTHESIZED_WIRE_8 & A1 & SYNTHESIZED_WIRE_10);

assign     Y2 = ~(A & SYNTHESIZED_WIRE_9 & SYNTHESIZED_WIRE_10);

assign     Y3 = ~(A & B & SYNTHESIZED_WIRE_10);


endmodule
```