GEBZE TEKNİK ÜNİVERSİTESİ

ELEKTRONİK MÜHENDİSLİĞİ

ELM235

LOJİK DEVRE TASARIM LABORATUVARI

LAB 0x3 Deney Raporu

Donanım Tanımlama Dillerine Giriş

Hazırlayanlar
1) 1801022022 – Alperen Karataş

1. Giriş

Bu deneyde donanım tanımlama dillerine giriş yapılmıştır. Modelsim'de SystemVerilog dilinin syntax yapısı öğrenilip gerekli kodlar yazılarak istenilen devreler oluşturulmaya çalışılmıştır.

2. Problemler

2.1. Problem I - Basit bir devre tasarımı ve simülasyonu

2.1.1. Teorik Araştırma

Bu problem için teorik bir araştırma bulunmamaktadır.

2.1.2. Deneyin Yapılışı

Soruda bize verildiği gibi A'B boolean denklemini sağlayan kod yazılmıştır. Testbench dosyası da oluşturulup kodu yazıldıktan sonra simülasyon aşamasına geçilmiştir.

Şekil 1

Şekil 1'deki simülasyon ekranında görüldüğü üzere devreye bütün giriş simülasyonları uygulanmıştır ve beklenen sonuç alınmışıtr.

2.1.3. Sonuçların Yorumu

A'B devresi yapı bakımından, devre mimarisi bakımından ve de kodunu yazabilme açısından zorlayıcı bir devre olmadığından kolaylıkla yazıp istenilen sonucu elde ettim. İlk problemin tek zorlayıcı kısmı programla yeni tanışmış olmanın getirdiği dezavantajlardı.

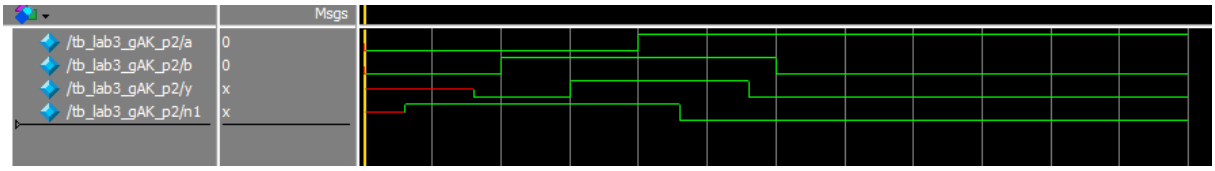
2.2. Problem II - Basit bir devrede gecikme simülasyonu

2.2.1. Teorik Araştırma

Bu problem için teorik bir araştırma bulunmamaktadır.

2.2.2. Deneyin Yapılışı

Burada problem 1'den farklı olarak sadece delay eklememiz istenmiş. 'timescale ibaresi kodun başına eklendi. NOT kapısı ve AND kapısına istenen delayler de koda eklendikten sonra simülasyon kısmına geçilmiştir.



Şekil 2

Şekil 2 de görüldüğü gibi giriş kombinasyonları uygulanmış ve beklenilen delay oluşmuştur. Kırmızı çizgiler unkown state'i ifade eder.

Y çıktısı, delayden kaynaklı olarak ilk başlarda unkown state durumundayken, sonradan devreye gerekli tepkiyi vermiştir. NOT ve AND kapılarına uygulanan delayler, y çıktısının simülasyon ekranına yansıttığı tepkilerde de delaye sebep olmuştur.

2.2.3. Sonuçların Yorumu

Devrenin kodunun yazımı yine devrenin kolaylığı göz önünde bulundurulduğunda çok zor değildi. Simülasyon ekranında başlangıçta görünen kırmızı çizginin sebebini anlamak biraz uğraştırdı. Daha sonra onun timescale belirlemekle ilgili olabileceğini öğrendim. Bir diğer bakış açısıyla kırmızı çizgilere delaylerin sebep olduğunu düşünüyorum.

2.3. Problem III – Glitch simülasyonu

2.3.1. Teorik Araştırma

Glitchin ne olduğu araştırılarak hatırlandı.

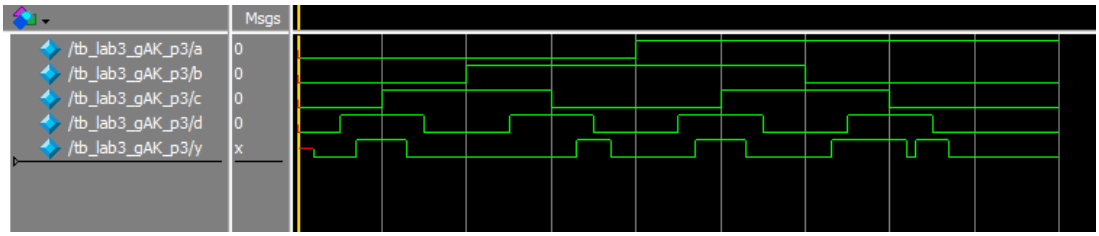
2.3.2. Deneyin Yapılışı

Verilen denklemin devre kodu yazıldı. Testbenchte ise grey code sıralamasına uygun olarak bütün komabinasyonlar denendi. Bunun sonucunda ilk başlarda herhangi bir glitch oluşmazken sonlara doğru glitch saptandı.

CD/AB	00	01	11	10	
00	0	0	0	0	
01	1	1	1	1	**(8)**
11	0	0	0	1	**(12)**
10	0	0	0	1	

Şekil 3. K-Map

NOT : Mavi renkli sayılar bir bubble'ı ifade ederken; kırmızı renkli rakamlar bir bubble'ı ifade eder.



Şekil 4

Şekil 3'teki K-Map'i incelediğimizde 1011 durumundan 1001 durumuna geçişte, yani C'nin 1 den 0'a düşüşünde bir glitch oluşacağını öngörebiliriz. Simülasyon ekranında da tam olarak bunu görüyoruz. Bunu ortadan kaldırmak için (8) ve (12) numaralı kutucukları bir bubble içine alırsak yeni bir kapı ekleyerek bu glitchi ortadan kaldırmış oluruz. Bu kutu içine aldığımız bubble, devreye AB'D içeren bir AND kapısı ekleyeceğimiz manasına gelir.

2.3.3. Sonuçların Yorumu

Devre kodu denklemde verildiği üzere yazıldı ancak asıl önemli noktanın testbench dosyasının yazımında olduğunu anladım. İlk başta testbenchteki bütün giriş kombinasyonlarını binary kod dizilimine göre denedim. Bu dizilim sonucu simülasyon ekranında herhangi bir glitch oluşmadığını gördüm. Daha sonra K-map'i dikkatli bir şekilde inceleyerek giriş kombinasyonları sırasını grey code'a göre düzenlemeye karar verdim. K-Map'ten de öngördüğüm gibi tam olarak beklediğim noktada, 1011→1001 geçişinde glitchi saptadım. Kodu yazmaktan daha zorlayıcı olan kısım glitch oluşup oluşmayacağını; oluşursa hangi geçişte oluşacağını anlamaya çalışmaktı.

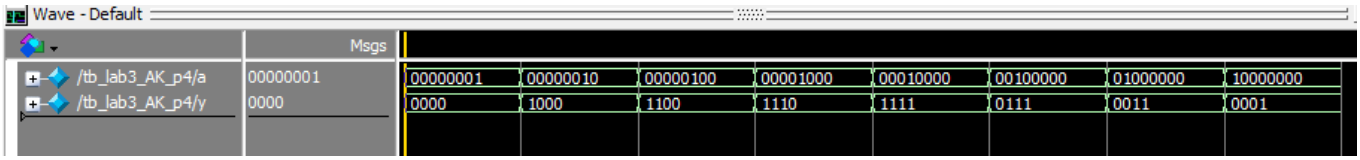
2.4. Problem IV - Çözücü tasarımı

2.4.1. Teorik Araştırma

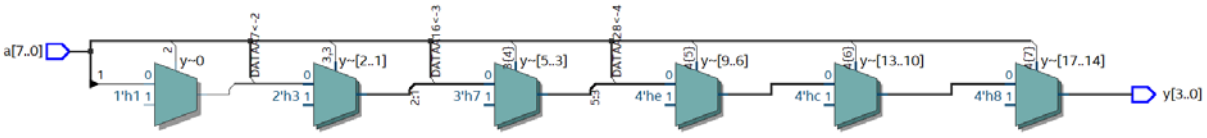
Çözücünün ne olduğu araştırılarak hatırlandı.

2.4.2. Deneyin Yapılışı

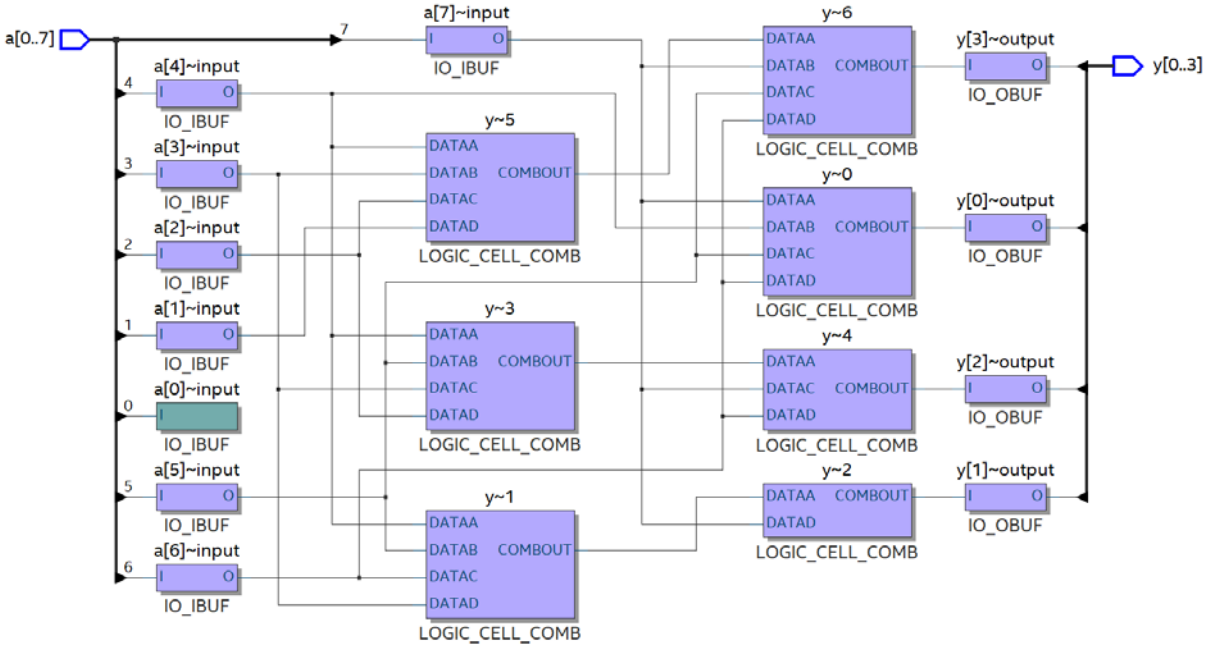
Anlatılan durum conditional assignments kullanılarak kodlandı. Sonrasında ise simülasyon çıktısı elde edildi. Kod A şıkkı için yazıldı; yani devre A şıkkına göre tasarlandı. Simülasyon çıktısı alındı.



Şekil 5. A şıkkı için simülasyon çıktısı



Şekil 6. A şıkkı için RTL şeması



Şekil 7. A şıkkı için Post Fitting

Flow Summary

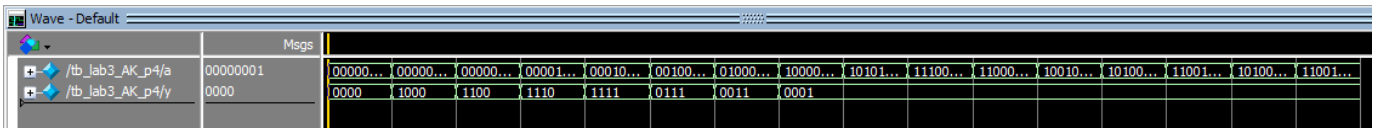
<<Filter>>

Flow Status	Successful - Tue Apr 07 03:41:32 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	problem4
Top-level Entity Name	lab3_AK_p4
Family	MAX 10
Device	10M08DAF484C8G
Timing Models	Final
Total logic elements	8 / 8,064 (< 1 %)
Total registers	0
Total pins	12 / 250 (5 %)
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 8.

Devrenin ne kadar yer kapladığı Şekil 8'de görülmektedir.

B şıkkına göre ise 16 kombinasyonlu simülasyon çıktısı Şekil 9'da görüldüğü gibidir.



Şekil 9

D şıkkına gelecek olursak;

Dijital devreler için test oluşturmak için iki yöntem vardır. Yüksek düzeyli (RTL) açıklamaların kod kapsamını amaçlayan EFSM tabanlı bir yöntem ve düşük düzeyli (kapı) açıklamaya dayalı bir denklik kontrolü. Üretilen testler için yüksek seviye kodu ve düşük seviye arıza kapsamı ölçülür. Birkaç arıza modeli için düşük seviyeli mutantlar üretildi. Sonuçlar, çoğu durumda, mutant kapsamının RTL testleri için oldukça düşük olduğunu göstermektedir. Tersine, düşük seviyeli testler, yüksek seviyeli testlerle daha düşük veya aynı RTL kodu kapsamına sahiptir.

2.4.3. Sonuçların Yorumu

A şıkkının kodu yazıldı ve beklenen sonuç simülasyon ekranında göründü. B şıkkına gelindiğinde, devre için testbenchte belirtilen ilk 8 kombinasyona eklenen kombinasyonlar, devrenin en son verdiği çıktının devamını simülasyon ekranında göstermeye devam etmiştir. Yani 8. giriş kombinasyonu 10000000 iken çıktı 0001'dir ve bundan sonraki giriş kombinasyonları için de çıktı 0001 olmaya devam etmiştir. Denenen 16 girişli kombinasyon aşağıdadır:

a= 8'b00000001 ; #10

a= 8'b00000010 ; #10

a= 8'b00000100 ; #10

a= 8'b00001000 ; #10

a= 8'b00010000 ; #10

a= 8'b00100000 ; #10

a= 8'b01000000 ; #10

a= 8'b10000000 ; #10

a= 8'b10101010 ; #10

a= 8'b11100000 ; #10

a= 8'b11000100 ; #10

a= 8'b10010100 ; #10

a= 8'b10100100 ; #10

a= 8'b11001001 ; #10

a= 8'b10100100 ; #10

a= 8'b11001001 ; #10

Bu simülasyon sonucunun belirttiğim gibi çıkmasının nedeni kombinasyon diziliminin yukarıdaki gibi olmasıdır. Farklı dizilimlerde farklı simülasyon görüntüleri ortaya çıkabilir.

2.5. Problem V - Yapısal tasarım (Structural Design)

2.5.3. Teorik Araştırma

Ders slaytlarından Structural Design araştırması yapılmıştır.

2.5.4. Deneyin Yapılışı

İlk etapta 8 bitlik 2x1 mux tasarlandıktan sonra, bu 2x1 muxlar kullanılarak 4x1 mux tasarımı yapılmıştır.

2x1 muxlar kullanılarak yapılan 4x1 mux tasarımının simülasyon sonucu aşağıda görülmektedir.

Şekil 10

E şıkkı için;

Yapısal tasarımın avantajları:

- Okuması ve anlaması daha kolaydır.
- Kullanıcı dostudur.
- Görünüş açısından daha anlaşılırdır.
- Makine tabanlı olmak yerine temeldeki probleme dayalıdır.
- Daha az çaba ve zaman gerektirdiğinden geliştirmesi daha kolaydır.
- Hata ayıklama daha kolaydır.
- Çoğunlukla makineden bağımsızdır.

2.5.5. Sonuçların Yorumu

Structural Design kullanılarak yapılan B şıkkındaki çalışmanın doğru sonuçlar verip vermediğinden emin değilim. Şekil 10'da görüldüğü üzere özellikle y çıktısı unkown state olarak gözüküyor. Ne kadar uğraşırsam uğraşayım bu durumu düzeltemedim. Bunların üstüne Modelsimde compile sonucu hatasız gözüken kod bloğu, Quartus Prime programında bir dizi compile hatasına sebep oldu ve yine bütün çabalarıma rağmen bu sorunun üstesinden gelemedim. Bu yüzden RTL, PostFitting şemalarını rapora ekleyemedim. Deneydeki bütün problemler arasında uğraştırıcılık açısından en sıkıntılı problem bu olabilir.

3. Sonuçlar ve Genel Yorumlar

Genel itibariyle oldukça verimli bir o kadar da yorucu bir lab çalışmasıydı. Kod ile devre tasarımı giriş açısından zorlayıcıydı. Daha önce bizzat çizerek tasarladığımız devreleri, kodlayarak tasarlamak ve bunun çıktısını simülasyon ekranında görmek (bir o kadar da görememek) donanım dünyasına girişte bizler için önemli bir adımdı. Problemlerin bazıları oldukça yorucu geçti. Örneğin simülasyon ekranında glitchi yakalayabilmek veya structural design problemi oldukça zordu.

4. Referanslar

1- Ders slaytları

2-
https://www.researchgate.net/publication/312408069_Testing_logic_circuits_at_different_abstraction_levels_An_experimental_evaluation

3-

https://www.geeksforgeeks.org/structured-programming-approach-with-advantages-and-disadvantages/

Kodlar

Problem 1.

```
/* lab3_gAK_p1.sv
*
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 1
* Y = NOT A and B denkleminin
gerçeklemesi
*
*/

module lab3_gAK_p1 (
input logic a, b,
output logic y
);
assign y = ~a & b;
endmodule
```

```
/* tb_lab3_gAK_p1.sv
*
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 1 Testbench
* Y = NOT A and B denkleminin simulasyonu
* Bütün olası girişlere göre çıkış gözlemlenir.
*
*/

`timescale 1ns/1ps

module tb_lab3_gAK_p1 ();

logic a, b; // all the inputs
logic y; // all the outputs

lab3_gAK_p1 uut0(.a(a), .b(b), .y(y));

initial begin
a = 0; b = 0; #10; // a = 0, b = 0, wait 10
ns
b = 1; #10; // a = 0, b = 1, wait 10 ns
a = 1; #10; // a = 1, b = 1, wait 10 ns
b = 0; #10; // a = 1, b = 0, wait 10 ns
#20; // wait 20 ns after completion
$stop; // stop the simulation
end
endmodule
```

Problem 2.

```
/* lab3_gAK_p2.sv
*
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 2
* Y = NOT A and B denkleminin delay
uygulanarak gerçeklemesi
*
*/

`timescale 1ns/1ps

module lab3_gAK_p2 (
input logic a, b,
logic n1,
output logic y
);
assign #3 n1 = ~a;
assign #5 y = n1 & b;
endmodule
```

```
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 2 Testbench
* Delay uygulanmış Y = NOT A and B denkleminin
simulasyonu
* Bütün olası girişlere göre çıkış gözlemlenir.
*
*/


`timescale 1ns/1ps

module tb_lab3_gAK_p2 ();

logic a, b; // all the inputs
logic y; // all the outputs
logic n1;

lab3_gAK_p2 dut0(.a(a), .b(b), .y(y), .n1(n1));

initial begin
a = 0; b = 0; #10; // a = 0, b = 0, wait 10 ns
b = 1; #10; // a = 0, b = 1, wait 10 ns
a = 1; #10; // a = 1, b = 1, wait 10 ns
b = 0; #10; // a = 1, b = 0, wait 10 ns
#20; // wait 20 ns after completion
$stop; // stop the simulation
end
endmodule
```

Problem 3.

```
/* lab3_gAK_p3.sv
*
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 3
* Delay uygulanmış X = AB'C + C'D denkleminin gerçeklemesi
*
*/

`timescale 1ns/1ps

module lab3_gAK_p3 (

input logic a,b,c,d,
logic n1,n2,n3,n4,
output logic y
);

assign #2 n1 = ~b;
assign #2 n2 = a&n1&c;
assign #2 n3 = ~c;
assign #2 n4 = n3&d;
assign #2  y = n2|n4;

endmodule
```

```
/* tb_lab3_gAK_p3.sv
*
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 3 Testbench
* Y = NOT A and B denkleminin simulasyonu
* Bütün olası girişlere göre çıkış gözlemlenir.
*
*/

`timescale 1ns/1ps

module tb_lab3_gAK_p3 ();

logic a,b,c,d;
logic y;

lab3_gAK_p3 uut0(.a(a), .b(b), .c(c), .d(d), .y(y));

initial begin
a = 0; b = 0; c = 0; d = 0; #10;  // 0
a = 0; b = 0; c = 0; d = 1; #10;  // 1
a = 0; b = 0; c = 1; d = 0; #10;  // 2
a = 0; b = 0; c = 1; d = 1; #10;  // 3
a = 0; b = 1; c = 0; d = 0; #10;  // 4
a = 0; b = 1; c = 0; d = 1; #10;  // 5
a = 0; b = 1; c = 1; d = 0; #10;  // 6
a = 0; b = 1; c = 0; d = 1; #10;  // 7
a = 1; b = 0; c = 0; d = 0; #10;  // 8
a = 1; b = 0; c = 0; d = 1; #10;  // 9
a = 1; b = 0; c = 1; d = 0; #10;  // 10
a = 1; b = 0; c = 1; d = 1; #10;  // 11
a = 1; b = 1; c = 0; d = 0; #10;  // 12
a = 1; b = 1; c = 0; d = 1; #10;  // 13
a = 1; b = 1; c = 1; d = 0; #10;  // 14
a = 1; b = 1; c = 1; d = 1; #10;  // 15
#20;
$stop; // stop the simulation
end
endmodule
```

Problem 4.

```
/* lab3_AK_p4.sv
*
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 4
* Çözücü tasarımı
*
*/


module lab3_AK_p4(

input logic [7:0] a,
output logic [3:0] y
);
assign y=a[7]? 4'b0001 :a[6]? 4'b0011 :a[5]? 4'b0111 :a[4]?
4'b1111 :a[3]? 4'b1110 :a[2]? 4'b1100 :a[1]? 4'b1000 : 4'b0000
;

endmodule
```

```
/* tb_lab3_AK_p4.sv
*
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 4 Testbench
* Çözücü simülasyonu
* Bütün olası girişlere göre çıkış gözlemlenir.
*
*/

`timescale 1ns/1ps
module tb_lab3_AK_p4();
logic [7:0]a;
logic [3:0]y;

lab3_AK_p4 uut0(.a(a), .y(y));

initial begin
a= 8'b00000001 ; #10
a= 8'b00000010 ; #10
a= 8'b00000100 ; #10
a= 8'b00001000 ; #10
a= 8'b00010000 ; #10
a= 8'b00100000 ; #10
a= 8'b01000000 ; #10
a= 8'b10000000 ; #10

$stop;
end
endmodule
```

Problem 5.

A şıkkı

```
/* lab3_AK_p4.a.sv
*
* Hazırlayanlar:
* Furkan Çaycı
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 1
* 8 bitlik 2x1 mux tasarımı
*
*/
module mux2 (

input logic [7:0] d0, d1,
input logic s,
output logic [3:0] y
);
assign y = s ? d1 : d0;


endmodule
```

B şıkkı

```
/* lab3_AK_p5.sv
*
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 5
* (8 bitlik) 2x1 mux kullanarak 4x1 mux tasarımı
*
*/

module mux2_1 (
input logic [7:0] a, b,
input logic s0,
output logic [7:0] x
);
assign x = s0 ? a : b;

endmodule


module mux2_2 (
input logic [7:0] c, d,
input logic s0,
output logic [7:0] z
);
assign z = s0 ? c : d;

endmodule

module mux2_3 (
input logic [7:0] e, f,
input logic s1,
output logic [7:0] y
);
assign y = s1 ? e : f;

endmodule


module mux4 (

input logic [7:0] a, b, c, d, e, f,
input logic s0, s1,
logic [7:0] n1,n2,
output logic [7:0] y
);
mux2_1 inst0(.a(a), .b(b), .x(n1));
mux2_2 inst1(.c(c), .d(d), .z(n2));
mux2_3 inst2(.e(n1), .f(n2), .y(y));

endmodule
```

B şıkkı için testbench

```
/* tb_lab3_AK_p5.sv
*
* Hazırlayanlar:
* Alperen Karataş
*
* Notlar:
* ELM235 2020 Bahar Lab3 - Problem 5 Testbench
* (8 bitlik) 2x1 mux kullanarak 4x1 mux tasarımı simülasyon deneyi
* Bazı olası girişlere göre çıkış gözlemlenir.
*
*/

`timescale 1ns/1ns

module tb_mux4 ();

logic [7:0] a, b, c, d, e, f;
logic s0, s1;
logic [7:0] n1,n2;
logic [7:0] y;

mux4 dut0( .a(a), .b(b), .c(c), .d(d), .s0(s0), .s1(s1), .y(y) );

initial begin

a={8'b00000001}; #10
b={8'b00000000}; #10
c={8'b00000001}; #10
d={8'b00000010}; #10
s0=0; #10
s1=0; #10
$stop;
end
endmodule
```