Laboratuvar Çalışması 0x3

Donanım Tanımlama Dillerine Giriş

I. Amaç

Bu labın amacı

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

II. Uyarılar

1. Lab'a başlamadan önce ders kadrosu tarafından verilen **ModelSim ve Quartus Prime kullanımı** adlı eğitim dökümanını takip edin, ve orada yapılan aşamaları uygulayabildiğinizden emin olun.
2. Lab içerisinde geçen HDL kelimelerini derste hangi dil seçilmişse o dili kullanarak değiştirin. (VHDL, SystemVerilog, ...)
3. Dosya adlarini her zaman küçük harflerle **labX_gY_pZ.uzantı** olarak kaydedin.
    a. X yerine lab numarasını, Y yerine grup numaranızı, Z yerine problem numarasını ekleyin.
    b. Mesela lab3, SystemVerilog, Problem 1 ve grup numaranız 18 ise, **lab3_g18_p1.sv** olarak kaydedin.
    c. Devre adını da **lab3_g18_p1** olarak belirleyin.
    d. Testbench isimlerinin başına **tb_** önekini koyun, ve aksi belirtilmedikçe girişler arası bekleme süreleri için 10 ns kullanın.
4. Rapor hazırlarken bu dokümanın sonundaki **EK bölümünde olduğu** gibi bir **tablo içerisine mono fontlarla** (Courier New, Roboto Mono, Consolas, vbg.) **kodlarınızı** ekleyin. Kodlarınızı ayrı bir dosya olarak **göndermeyin**. Resim olarak eklemeyin, ve formatlamanıza özen gösterin.
5. Ek bölümünde verildiği gibi bir **şablonu** her **kodunuzun başına** ekleyin.

III. Problemler

Problem 1 - Basit bir devre tasarımı ve simülasyonu

$$Y = \overline{A}\ B$$

Denklem 1

A. Denklem 1 deki Boolean denklemini HDL ile tasarlayin.

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.

Problem 2 - Basit bir devrede gecikme simülasyonu

A. Problem 1 deki devreleriniz kopyalayın, NOT kapısı sonucu için dahili bir sinyal oluşturun, ve NOT kapısını 3ns, AND kapısını 5 ns gecikme ile atayın. Bunun için devrenin başına **timescale** eklemeniz gerekmektedir. Bunun için her bir kapıyı ayrı parçalara bölmeniz gerekmektedir. Aşağıda bir örneği verilmiştir.

```
assign #3 n1 = ~a;
assign #5 y  = n1 and b;
```

B. Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemleyin.
C. Çıkan dalga şemasını yorumlayın.

Problem 3 - Glitch simülasyonu

$$X = A\,\overline{B}\,C + \overline{C}\,D$$

Denklem 2

A. Denklem 2 deki Boolean denklemini HDL kullanarak tasarlayın. Her bir kapıya **2ns** gecikme vererek atama yapın (NOT, AND ve OR)
B. Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemleyin. Her bir giriş ataması arasında **10ns** bekleyin. Burada glitch oluşup oluşmadığı hakkında bilgi verin.
C. Devrenin K-Mapine bakarak, hangi koşulda glitch oluşabileceği hakkında yorum yapın, ve bulduğunuzu o koşulu Testbench te test edin.

Şekil 1. Glitch gözlemi. Şekil verilen farklı giriş kombinasyonlarına göre çıkış sinyalini göstermektedir. Başlangıçta tüm giriş paternlerine karşı bir glitch oluşmadığını, fakat daha sonra özel bir giriş kombinasyonu ile glitch olduğunu görebiliriz. (En sondaki kısa 1→0→1 dönüşü)

D. Çıkan dalga şeklini yorumlayın, ve glitchi ortadan kaldırmak için nasıl bir devre eklenmesi gerektiğini belirtin.

Problem 4 - Çözücü tasarımı

A. 8-girişli bir devrenin, verilen isterlere uygun girişlerinin önceliğini belirleyen bir devre tasarlayınız. Girişiniz **8-bitlik bir bus** olsun. Çıkışınız **4-bitlik bir bus** olsun. Girişin 8. elemanı (i.e. `a[7]`) aktif ise, çıkış 0001, 8. girişi aktif değil ve 7. girişi aktif ise (i.e. `a[7]' a[6]`) çıkış 0011, gibi çıkış sırası **Grey Code**[1] ile kodlanmış bir devreyi sadece **Şartlı Atamalar** (*Conditional Assignments)* kullanarak tasarlayınız.

B. Devrenin simülasyonunu yaparak, rastgele seçtiğiniz 16 farklı giriş kombinasyonuna göre çıkış dalga şeklini gözlemleyiniz.
C. Devreyi Sentezleyerek RTL ve Post-Fitting şemalarını ekleyiniz, ne kadar yer kapladığı hakkında bilgi veriniz.
D. Devreyi oluşturmanın **alternatif bir yolundan bahsederek**, devreyi **daha yüksek seviye soyutlamada** tanımlamanın avantajları veya dezavantajları hakkında **yorum** yapınız.

Problem 5 - Yapısal tasarım (Structural Design)

A. 8-bitlik 2-ye-1 seçici devresi tasarlayın.
B. A daki devreyi kullanarak 8-bitlik 4-e-1 seçici devre tasarlayın.
C. B deki devrenin simülasyonunu yaparak, doğruluğunu gözlemleyin.
D. B deki devreyi Sentezleyerek RTL ve Post-Fitting şemalarını ekleyiniz, ne kadar yer kapladığı hakkında bilgi veriniz.
E. Yapısal Tasarımın avantajları hakkında yorum yapınız.

IV. Referanslar

---

[1] https://en.wikipedia.org/wiki/Gray_code

V. Ekler

EK A - Problem 1 HDL kodları

```
/* lab3_g0_p1.sv
 *
 * Hazırlayanlar:
 *   Furkan Çaycı
 *
 * Notlar:
 *   ELM235 2020 Bahar Lab3 - Problem 1
 *   Y = NOT A and B denkleminin gerçeklemesi
 *
 */

module lab3_g0_p1 (
    input  logic a, b,
    output logic y
);

    assign y = ~a & b;

endmodule
```

```
/* tb_lab3_g0_p1.sv
 *
 * Hazırlayanlar:
 *   Furkan Çaycı
 *
 * Notlar:
 *   ELM235 2020 Bahar Lab3 - Problem 1 Testbench
 *   Y = NOT A and B denkleminin simulasyonu
 *   Bütün olası girişlere göre çıkış gözlemlenir.
 *
 */

// Time Units and resolution of the simulation
`timescale 1ns/1ps

module tb_lab3_g0_p1 ();

    // These names do not need to be the same as the part1 ports
    // but, we make them the same to make it easier to understand
    // which is connected to what port
    logic a, b; // all the inputs
    logic y;    // all the outputs

    // Explicit port mapping. Always prefer it, vs. the implicit.
    lab3_g0_p1 uut0(.a(a), .b(b), .y(y));

    // This part is applied to the circuit sequentially.
    // The results can be inspected
    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
```