Laboratuvar Çalışması 2

Donanım Tanımlama Dilleri Uygulamaları: Birleşik Devreler

A. Amaçlar

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,

B. Bilgilendirme

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. Dosya adlarini her zaman küçük harflerle **labX_gY_pZ**.uzantı olarak kaydedin. **X** yerine lab numarasını, **Y** yerine grup numaranızı, **Z** yerine problem numarasını ekleyin. Örneğin lab3, SystemVerilog, Problem 1 ve grup numaranız 18 ise, **lab3_g18_p1.sv** olarak kaydedin. Devre adını da **lab3_g18_p1** olarak belirleyin. Test tezgah (Testbench) isimlerinin başına **tb_** önekini koyun, ve aksi belirtilmedikçe girişler arası bekleme süreleri için 10ns kullanın.
4. Rapor hazırlarken bu dokümanın sonundaki EK bölümünde olduğu gibi bir metin kutusu içerisine mono fontlarla (Courier New, Roboto Mono, Consolas gibi) 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.

C. Teslim Edilecekler

1. Laboratuvar raporu **tek bir PDF dosyası** halinde olacak ve **farklı bir formatta teslim ettiğiniz raporlar geçersiz sayılacaktır.** İçerik olarak Lab0 daki örnek rapora bağlı kalınız. Tüm devre şemalarını ve simülasyon sonuçlarını gösteren dalga şekillerini ve zamanlama diyagramlarını eklemeyi unutmayınız.

D. Problemler

Problem 1 - İstenmeyen Darbe Sinyalleri: Glitchler

Denklem 1: **Y = AB'C + C'D**

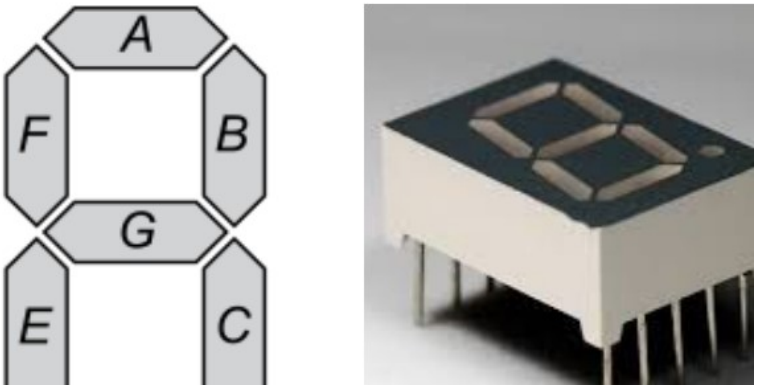
A. Denklem 1'de verilen Boole denklemini Donanım Tanımlama Dili (DTD) kullanarak bir modül olarak gerçekleyin. Her kapıya 2ns gecikme vererek atama yapın (propagasyon gecikmesini modellemiş oluyoruz).
   Örnek: Y = A'B Boolean denklemi için bır NOT ve bir AND kapısı gerekiyor. Modül içinde
   ```
   assign #2 n1 = ~a;
   assign #2 y = n1 & b;
   ```
   ifadeleriyle gecikmeleri modellemiş oluruz. `timescale direktifini de DTD kodunun başına ekleyiniz.

B. Tasarladığınız DTD modelini ve ilgili fonksiyonel test tezgahı kodunu girişlere bütün olası kombinasyonları her 10ns lik adımlarda uygulayacak ve çıkış sinyalini gözlemleyecek şekilde DTD ile yazarak simüle ediniz.
C. Elde ettiğiniz zamanlama diyagramını kodlarınızla birlikte raporlayınız ve yorumlayınız. Glitch durumu hakkında bilgi veriniz.
D. Denklem 1'de verilen Boole denkleminin doğruluk tablosunu oluşturup, K-map üzerinde hangi şartlarda glitch oluşabileceği hakkında yorum yapınız. Bulduğunuz koşulu yazdığınız test tezgah kodu ile test ediniz.

Şekil 1. Örnek glitch gözlemi. Şekil verilen farklı giriş kombinasyonlarına göreçıkış 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)

E. Glitch gözlemlerseniz bunu gidermek için ne yapmak gerektiğini belirterek yazdığınız DTD modülünü güncelleyerek yeniden test edin ve glitch oluşmadığını gösterin.

Problem 2 – Çözücüler (Decoder) ve Uygulamaları



Şekil – 2: 7 bölmeli LED gösterge.

Şekil 2'de 7 bölmeli LED ekran (7BLE) (7-segment LED display) gösterilmektedir.[1] Bu elemanın ABCDEFG bölmelerindeki LEDlere bağlı 7 tane girişi olduğunu ve her bir girişe lojik 1 gönderildiğinde kendi bölmesini LEDi yakacak şekilde tasarlandığını varsayın. Başka bir deyişle LEDlerin yanması için ortak pinlerinin lojik 0 olması gerektiğini (common cathode) varsayın.

A. Şekil 2'deki 7BLE elemanının bölmeleri Tablo 1'de verilen şekilde çalıştırmak için tasarlanacak 4 girişli bir çözücü (Decoder) devresinin doğruluk tablosunu oluşturunuz. Kullanılmayan giriş kombinasyonları için önemseme (don't care) kullanınız.

Girişler ($x_3x_2x_1x_0$)	Çıkışlar 7BLE Deseni
0000	-
0010	E
0011	3
0101	L
1100	C
1101	2
1110	5

Tablo-1: 7BLE çalışma tablosu.

B. Her bir çıkış için K-Map ve önemseme koşullarını kendi lehinize kullanarak en sade Boolean cebri ifadesini bulunuz. Toplamda kaç devre elemanı kullandığınızı (NOT kapıları dahil) belirtiniz.

C. Çözücü devresinin DTD modelini davranışsal (behavioral) tarzda yazınız ve Tablo-1'deki gibi çalışacak şekilde gerçekleyiniz.

D. İlgili fonksiyonel test tezgahı kodunu girişlere bütün olası kombinasyonları her 10 ns de bir uygulayacak ve çıkış sinyallerini gözlemleyecek şekilde yazarak simüle ediniz. Elde ettiğiniz zamanlama diyagramını kodlarınızla birlikte raporlayınız.

E. Devrenin ne kadar yer kapladığını (resource utilization report) sentezlenen RTL ve eşleştirme ardı devre şemalarını da ekleyerek karşılaştırmalı şekilde yorumlayınız.

F. Devrenin, sentezleme sonucunda ne kadar yer kapladığı hakkında bilgi veriniz, sentezlenen RTL ve eşleştirme sonrası devre şemalarını ekleyiniz. Problem 1'deki devre sentezi ile karşılaştırınız ve sonuçları yorumlayınız.

Referanslar

[1] https://en.wikipedia.org/wiki/Seven-segment_display

E. Ekler

1. Örnek bir SystemVerilog DTD tasarımı ve test tezgahı

```
/* lab1_g0_p10.sv
*
* Hazırlayanlar:
* Ihsan Cicek
*
* Notlar:
* ELM235 2023 Bahar Lab1 - Problem 10
* Y = NOT A and B denkleminin gerçeklemesi
*
*/
module lab1_g0_p10 (
input logic a, b,
output logic y
);
      assign y = ~a & b;
endmodule


/* tb_lab1_g0_p10.sv
*
* Hazırlayanlar:
* Ihsan Cicek
*
* Notlar:
* ELM235 2023 Bahar Lab1 - Problem 10 Testbench
* Y = NOT A and B denkleminin simulasyonu
* Bütün olası girişlere göre çıkış gözlemlenir.
*
*/
// Zaman birimi ve simulasyon cozunurlugu
`timescale 1ns/1ps
module tb_lab1_g0_p10 ();
// Test tezgahlarinda port bulunmaz

logic a_tb, b_tb; // test tezgahi giris sinyal tanimlari
logic y_tb; // test tezgahi cikis sinyal tanimlari

// Test edilecek modulun yaratimi ve port baglantilarinin yapilmasi
// mut = module under test,
lab1_g0_p10 mut0(.a(a_tb), .b(b_tb), .y(y_tb));

// Bu kisimda sinyaller test edilen devreye sirali olarak uygulanir.
// Sonuclar test edilen devre cikislarinda gozlenebilir.
initial begin
      a_tb = 0; b_tb = 0;
      #10; // a = 0, b = 0 yap, 10 ns bekle

      b_tb = 1;
      #10; // a = 0, b = 1 yap, 10 ns bekle
      a_tb = 1;
      #10; // a = 1 yap, b = 1, 10 ns bekle

      b_tb = 0;
      #10; // a = 1, b = 0 yap, 10 ns bekle

      #20; // simulasyondan sonra 20 ns bekle
      $finish; // sımulasyonu sonlandır
end
endmodule
```