Laboratuvar Çalışması 4

Senkron Lojik Tasarım ve Aritmetik Lojik Devreler

A. Amaçlar

1. Donanım tanımlama dillerini (DTD) kullanarak devre tasarımı yapmak,
2. Aritmetik devreleri yapısal formda hiyerarşik olarak tasarlamak
3. Sentezleyici araçları kullanarak, DTD ile tanımlanan aritmetik devreleri FPGA için sentezlemek.
3. Simülasyon araçları kullanarak, otomatik simülasyon yaptırmak,
4. Birleşik lojik devreleri senkronize etmek
5. Tasarımların fonksiyonel testlerini dosyadan okuma yaparak otomatik şekilde gerçekleştirmek ve sonuçları otomatik şekilde karşılaştırarak raporlamak.

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 (Test tezgahı) 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 – Bellek Elemanlarının Karşılaştırılması

DTD davranışsal ifadelerini kullanarak birer latch, yükselen kenar tetiklemeli D-FlipFlop ve düşen kenar tetiklemeli D-FlipFlop tasarlayın. Her birinin girişini aynı sinyale, çıkışlarını ise ayrı sinyallere bağlayın. Ekte verilene benzer bir test tezgahı yardımıyla devrelerin fonksiyonel testlerini yapın ve sonuçları yorumlayın. RTL ve Tech map şemalarını karşılaştırınız. Modern senkron ardışık lojik devre tasarımlarında neden latch elemanlarının kullanımından sakınıldığını açıklayın.

Problem 2 – Aritmetik Devrelerin Tasarımı

A. Bir Half-Adder devresini DTD yapısal ifadelerini kullanarak tasarlayın, ardından bu tasarımı yeniden kullanarak bir Full-Adder devresi tasarlayın ve bunun 5 kopyasını birleştirip 5-bitlik bir Ripple-Carry Adder tasarımını hiyerarşik olarak gerçekleyiniz.

B. Test tezgahı oluşturarak, 5-bitlik RCA devrenizi fonksiyonel olarak doğrulayın. Tasarladığınız DTD modelini ve ilgili fonksiyonel test tezgahı kodunu girişlere 16 adet rastgele seçilmiş giriş değerlerini her 10ns lik adımlarda uygulayarak ve çıkış sinyalini gözlemleyerek ölçüm sonuçlarını raporlayınız. ModelSim dalga formu şemasını File→Export→Image.. ile PNG olarak çıkarabilirsiniz ya da tüm ModelSim projesinin ekran görüntüsü alabilirsiniz.

C. 5-bitlik RCA devresinin sentezleme sonuçlarını raporlayın ve yorumlayın.

Problem 3 – 32-bitlik Bir Aritmetik Lojik Birimin (ALU) Tasarımı

A. Tablo-1 de OPCODE detayları verilen 32-bitlik ve NZVC bayraklarını destekleyen bir ALU tasarımını davranışsal DTD ifadelerinden yararlanarak gerçekleyiniz.

op[2]	**operasyon**	**açıklama**
0000	addition	A + B
1000	subtraction	A - B
0001	shift left logical	A sinyalini B kadar sola kaydır[1]
0010	s compare	A sinyali B den büyükse 1, değilse 0 (signed comp)
0011	u compare	A sinyali B den büyükse 1, değilse 0 (unsigned comp)
0100	xor	A XOR B
0101	shift right logical	A sinyalini B kadar sağa kaydır[1] (0 padded)
1101	shift right arithmetic	A sinyalini B kadar sağa kaydır[1] (sign padded)
0110	or	A OR B
0111	and	A AND B

Tablo-1: ALU İşlem kodu (OPCODE) listesi

Not 1: B nin son 5 bitine bakarak kaydırma yapılacak. Geri kalan bitler ihmal edilebilir.
Not 2: Eğer aşağıdaki opcodelar haricinde bir giriş gelirse, hata biti oluşturulacak.

Örnek bir DTD modül kodu:

```
module alu (
        input  logic [31:0] a, b,
        input  logic [ 3:0] op,
        output logic [31:0] s,
        output logic n, z, v, c,
        output logic hata);
```

E. Ekler

1. Problem 1 için örnek bir SystemVerilog DTD tasarımı ve test tezgahı

```
/* tb_ornek.sv
 *
 * Hazırlayanlar:
 *   İhsan Çiçek
 *
 * Notlar:
 *   ELM235 2023 Bahar Lab4 test tezgahı örneği
 *
 */

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

module tb_ornek ();
    logic d, clk;
    logic q1, q2, q3;

    ornek uut0(...);

    // always blogu surekli calistirilacak
    // clock oluşturmak için clk sinyalini bir lojik
    // seviyeye çekip belli bir süre bekleyip, evirerek geri çekiyoruz
    // aşağıdaki haliyle 20ns lik peryotlu 50MHzlik bir clk oluşur.
    always
    begin
        clk = 0; #10; clk = 1; #10; //Duty cycle 50% yarım peryot 10ns
    end

    initial begin
        d = 0; #7; d = 1; #5; d = 0; #2;
        d = 1; #4; d = 0; #3; d = 1; #3;
        d = 0; #2; d = 1; #2; d = 0; #4;
        d = 1; #2; d = 0; #2; d = 1; #6;
        d = 0; #10;

        $stop;              // simülasyonu durdur ve beklet. Bitirmek için $finish;
    end

endmodule
```

2. Problem 2deki ALU Shift Operatörleri için carry oluşturma yöntemi