Laboratuvar Çalışması 5

Ardışık Lojik Devre ve Sonlu Otomat Tasarımı

A. Amaçlar

1. Donanım tanımlama dillerini (DTD) kullanarak devre tasarımı yapmak,
2. Ardışık lojik devreleri tasarlamak
3. Sentezleyici araçları kullanarak, DTD ile tanımlanan ardışık devreleri FPGA için sentezlemek.
4. Simülasyon araçları kullanarak, otomatik simülasyon yaptırmak,
5. Devrelerin çalışma prensiplerine ve isterlere göre çalıştığını doğrulamak için senaryo oluşturup onları test edebilmek.

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 – Yukarı serbest sayıcı devresi

Bu problemde devre aktif iken (en sinyali = 1) saatin her yükselen kenarında 0 dan verilen 5-bitlik bir prescaler (psc) değerine kadar sayıp, her psc değerine ulaştığında bir tick sinyali oluşturacak bir sayıcı devre tasarlayacaksınız. Devre aktif değilken (en sinyali = 0) sayaç değeri değişmemelidir. Ayrıca tasarımınıza bir senkron active-low reset sinyali ekleyip, reset geldiğinde sayacın sıfırlanabilmesini sağlayınız. Örnek bir modülün portları aşağıda verilmiştir. Modülünüz için basit bir testbench oluşturup, birkaç farklı psc değerlerine göre test ediniz.

```
module p1 (
       input logic clk, reset, en,
       input logic [4:0] psc,
       output tick);
```

Problem 2 – Yavaşlatılabilir aşağı sayıcı tasarımı

Bu problemde, aktif edildiğinde (en sinyali = 1) verilen 16-bitlik reload değerinden 0 a kadar azalarak sayıp, 0 a ulaştığında bitti sinyali oluşturan bir sayıcı devre tasarlayacaksınız. Ayrıca bir senkron reset sinyali ekleyip, reset geldiğinde counterınızı sıfırlayınız. Bu devreyi tasarlarken, Problem 1 de tasarladığınız devreyi instantiate edip, sayacınızı her tick geldiğinde bir düşürünüz.

- Tasarımınızı yaparken senkron tasarım prensiplerine bağlı kalınız.
- Active-low reset kullanınız.
- Sayaç değeriniz, devre aktifse ve Problem 1 deki devredeki tick sinyali geldiğinde düşürülecektir.
- Reload değeriniz 0 dan farklı ise ve devre aktif değilse (en sinyali = 0) reload değeri counterınıza atanacak.
- Eğer devre aktif ise reload değerinin değişmesi sayacınızı etkilemeyecek
- Bu devreye verdiğiniz psc değeri Problem 1 de tasarladığınız devreye iletilecek
- Devre aktifse ve counter 0 a ulaşmışsa, bitti sinyali verilecek (bitti = 1)
- 0 a ulaştıktan sonra o anki reload değeri countera yüklenip aynı şekilde çalışmaya devam edecek.

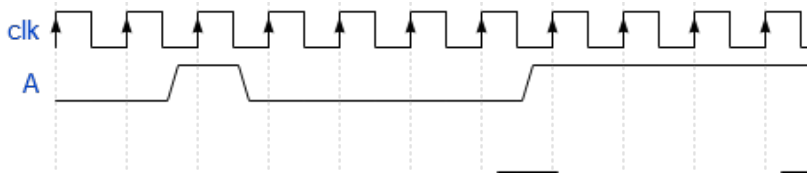
Örnek olarak psc 5 ve reload değeriniz 16 ise devreyi aktif ettiğiniz zaman, her 5 cycle da bir tick sinyali oluşacak ve her tick sinyalinde counter değeriniz 16 dan bir düşecek. Dolayısıyla 80 clock cycle sonra (16*5) bitti sinyali görmeniz lazım. (+- 1 cycle farklılık gösterebilir)

```
module p2 (
      input  logic clk, reset, en,
      input  logic [ 4:0] psc,
      input  logic [15:0] reload,
      output logic [15:0] cnt,
      output logic bitti
      );
```

Bu devreyi test etmek için oluşturacağınız testbench, farklı giriş kombinasyonlarını göz önünde bulundurmalıdır. Örnek olarak isterlerde belirtilen normal operasyon dışı reload değeri en sinyali aktif iken değişebilir. Bu ve bunun gibi durumda devrenin normal çalıştığını gözlemlemeniz gerekmektedir. Raporunuza testbench oluştururken düşündüğünüz senaryoları ve onlara karşı devrenin nasıl çalıştığını da ekleyiniz.

Problem 3 – Sonlu durum makinesi (FSM) tasarımı : bit deseni dedektörü

Bir girişi (A), bir çıkışı (Y) olan bir devrede, giriş sinyalinden 4 kere ardarda 0 veya 4 kere ardarda 1 geldiğinde çıkışı (Y) 1 yapan bir FSM devresi tasarlayın. Eğer 4 den daha fazla tekrar varsa, çıkış 1 kalmaya devam etmesi gerekmektedir. Şekil 1 de zamanlama örneği verilmiştir.



Şekil 1

A. Durum geçiş diyagramını (State Transition Diagram) çizin

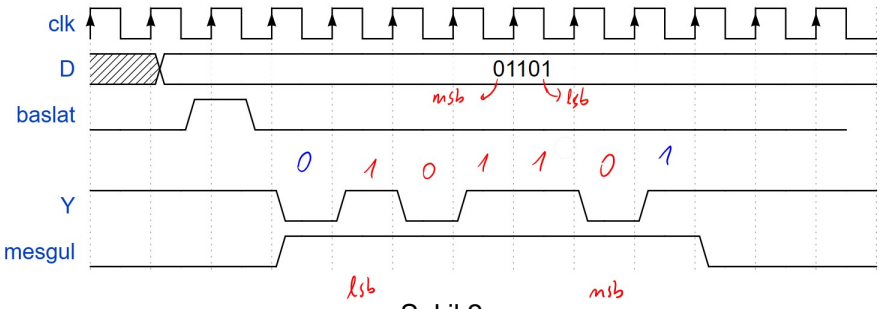
B. Bu diyagrama göre devrenizi DTD ile tasarlayıp, test edin.

Problem 4 – Sıralı Bit Pattern Üreteci

Bu problemde **5-bit D** değerini **baslat** geldiğinde aşağıdaki isterlere uygun olarak **Y** çıkışından göndermeniz istenmektedir. (normal bir shift register olarak çalışacağını düşünebilirsiniz.)

- Y çıkışı, baslat girişi 0 olduğu zaman lojik 1 olarak sürülecektir. FSM in S0 stateti burası olsun.
  - baslat girişi sadece 1 clock cycle süreyle aktif hale gelebilir.
- baslat 1 olduktan sonra önce 1 clock cycle lık lojik 0 gönderilecek
- sonrasında D nin LSB bit inden başlayarak her clock cycle da bir D nin bir sonraki biti gönderilecek.
- D nin bütün bitleri bittikten sonra 1 clock cycle lık lojik 1 gönderilecek
- FSM S0 statetine geri dönecek
- Devre S0 stateti haricindeki bütün state lerde **mesgul** çıkışı lojik 1 olarak sürülecek, S0 statetinde **mesgul** çıkışı lojik 0 olarak sürülecektir.
- İlk baştaki lojik 0 ı göndermek için bir state belirleyin.
- En sonraki lojik 1 i göndermek için bir state belirleyin.
- Aradaki D sinyallerini göndermek için state veya stateler belirleyin.

Örnek olarak D sinyalinin 01101 olduğunu varsayalım. start Y ve mesgul sinyallerinin zamanlama şeması Şekil 2 de verilmiştir.



Şekil 2

Not: D girişinin ne olduğunun önemi yoktur. Önemli olan gönderilen bitlerin sırasıdır.

```
module p4 (
  input  logic       clk, reset, en,
  input  logic [4:0] D,
  input  logic       baslat,
  output logic       Y,
  output logic       mesgul
);
```

A. Durum geçiş diyagramını (State transition diagram) çiziniz.

B. Bu diyagrama göre devrenizi DTD ile tasarlayıpd, devrenizi farklı veri kombinasyonlarıyla test ederek çalıştığını gözlemleyin.

E. Ekler

Ek bulunmamaktadır.