SIRALI MANTIK UYGULAMALARI

1. GİRİŞ

Bu derste, sıralı mantık içeren çeşitli basit devreleri Verilog ile Davranışsal Modelleme yaparak gerçekleştireceksiniz.

2. SIRAYLA YANAN LED IŞIKLAR

Bu bölümde, yan yana sıralanmış olan LED ışıkları sıralı olarak yakan bir devre gerçekleştirmeniz istenmektedir. Başlangıçta, 8 adet LED'ten sadece ilki yanacak, diğerleri yanmayacaktır. Bir sonraki çevrimde ise yanmakta olan LED'in yanındaki LED yanacak ve önceki LED sönecektir. Örneğin; her bir LED için mantık-0 ile yanmama, mantık-1 ile yanma durumunu gösterecek olursak, başlangıçta LED'ler 0000_0001 durumunda olacaktır. Bir sonraki çevrimde ise 0000_0010 durumuna geçeceklerdir. Her çevrimde bir sola kayanak, en soldaki LED yanana kadar işlem bu şekilde devam edecektir. En soldaki LED yandıktan sonra ise benzer şekilde saatin her yükselen kenarında yanmakta olan LED'in sağındaki LED yanacak ve eskisi sönecektir. Yani LED'ler, 0000_0001, 0000_0010, 0000_0100, ..., 1000_0000, 0100_0000, 0010_0000, ... şeklinde devamlı olarak durum değiştirecektir.

- **[Gerçekleştirme]** Verilog ile Davranışsal Modelleme yaparak gerçekleştireceğiniz LED'leri sırayla yakan modülün ismi "siraliLED" olmalıdır. Bu modülün, 8 bitlik "ledler" isminde çıkışı ve "clk" isminde saat girişi olacaktır. Modül saatin **her alçalan kenarında** işlem yapacaktır.
- **[Simülasyon]** Gerçekleştirmiş olduğunuz "siraliLED" modülünün LED çıkışlarına sırayla mantık-1 verdiğinden emin olmak için, bu modülü test eden "tb_siraliLED" adlı testbench kodunu yazın. Vivado yazılımını kullanarak modülün simülasyonunu yapıp, karedalga (Waveform) görünümünden modülün doğru çalıştığını kontrol edin.
- **[FPGA Kartı ile Deneme]** Simülasyon yaparak doğru çalıştığından emin olduğunuz "siraliLED" modülünü kullanarak aşağıda belirtilen maddeleri sağlayan "FPGA_siraliLED" isimli bir uygulama yazınız. Uygulamanızda,
  - 1 bitlik "clk" girişi bulunmalıdır.
  - 8 bitlik "ledler" çıkışı bulunmalıdır.
  - "ledler" çıkışını en sağdaki 8 LED'i kullanarak gösteriniz.
  - "clk" girişine bir saat bağlayınız. Kullandığımız FPGA kartının saat PIN'i W5'tir. Bir saatin kısıt dosyasında nasıl oluşturulabileceği aşağıdaki şekilde gösterilmiştir.

```
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -period 10.00 -waveform {0 5} [get_ports clk]
```

Şekil 1: Kısıt dosyasında saat üretilmesi

Uygulamanız için bir Bitstream üretin. FPGA'in LED'lerinin sırayla yandığını gözlemleyin.

3. AYARLANABİLİR SAYAÇ

Bu bölümde 6 bitlik ayarlanabilir sayaç modülü gerçekleştirmeniz istenmektedir. Bu sayacın kaç sayı atlayarak saydığı ve artan olarak mı yoksa azalan olarak mı saydığı ayarlanabilir olacaktır. Ayrıca sayacın değerini sıfırlayabilmek için modülün bir de *reset* denetimi olacaktır. Saatin her alçalan kenarında çalışacak olan bu modül, başlangıçta 0'dan bir artacak şekilde saymaya başlayacaktır. Modülün sayma işlemi gerçekleştirmesi veya değerini koruması kararını veren bir "etkin" girişi olacaktır. Bunun dışında, artan veya azalan olarak sayma ve kaç sayı atlayarak sayma ayarını yapmak için girişler olacaktır. Bu girişler yalnızca "ayar_yap" girişi mantık-1 olduğunda etkin olacaktır. Yani "ayar_yap" girişi mantık-0 olduğunda girişlerin gösterdiği şekilde ayarlama yapılmayıp, varolan durum korunacaktır. Sayma işlemi yapılırken 6 bit ile gösterilebilecek en büyük veya en küçük sayıya ulaşıldığında sayma işlemi sonlanacak ve sayaç sabit olarak bu en büyük veya en küçük değeri gösterecektir.

- **[Gerçekleştirme]** 6 bitlik ayarlanabilir sayaç modülüne "ayarliSayac" ismini verin. Bu modülün "clk" isminde bir saat girişi olacak ve modül saatin **her alçalan kenarında** işlem yapacaktır. "rst" ismindeki başka bir giriş sinyali de sayacı sıfırlamak için kullanılacaktır (rst == 1'b1 olduğunda sayaç sıfırlanacaktır). "etkin" girişi ise sayacın saymaya devam ettiğini veya durakladığını gösterecektir (mantık-0 iken sayaç değerini koruyacaktır). Sayacın kaç sayı atlayarak ve hangi yönde (azalan veya artan) ayarlamak için "ayar_yap" girişi kullanılacaktır. "ayar_yap" girişi mantık-1 olduğunda 3 bitlik "sayma_miktari" ve tek bitlik "sayma_yonu" girişlerine göre sayaç ayarlanacaktır. Sayacın ayarı yapıldıktan sonraki saatin alçalan kenarında yeni ayara göre sayaç çalışır olacaktır. Sayacın değerini göstermek için 6 bitlik "out_sayac" çıkışı kullanılacaktır. Özellikleri verilen bu modülü Verilog ile davranışsal modelleme kullanarak gerçekleştirin.
- **[Simülasyon]** Gerçekleştirmiş olduğunuz "ayarliSayac" modülünün verilen girdilere göre artıp azaldığından emin olmak için, bu modülü test eden "tb_ayarliSayac" adlı testbench kodunu yazın. Vivado yazılımını kullanarak modülün simülasyonunu yapıp, karedalga (Waveform) görünümünden modülün doğru çalıştığını kontrol edin.
- **[FPGA Kartı ile Deneme]** Simülasyon yaparak doğru çalıştığından emin olduğunuz "ayarliSayac" modülünü kullanarak aşağıda belirtilen maddeleri sağlayan "FPGA_ayarliSayac" isimli bir uygulama yazınız. Uygulamanızda,
  - 1 bitlik "clk", "rst", "etkin", "ayar_yap" , "sayma yonu" girişleri ve 3 bitlik "sayma_miktari" girişi bulunmalıdır.
  - 6 bitlik "out_sayac" çıkışı bulunmalıdır.
  - "rst" girişini FPGA kartınızdaki orta buton ile, "sayma_miktarı" ve "sayma_yonu" girdilerini en soldaki 4 Switch ile, "etkin" ve "ayar_yap" girişlerini ise en sağdaki 2 Switch ile belirlenecek şekilde ayarlayınız.
  - "clk" girişine bir saat bağlayınız.
  - "out_sayac" çıkışını en sağdaki 6 LED'i kullanarak ikilik tabanda gösteriniz.

  Uygulamanız için bir Bitstream üretin. FPGA'in Switch'leri yardımıyla istediğiniz değerleri verin, LED'de yanan sayının verdiğiniz girdilere göre saydığını kontrol edin.

4. PARAMETRİK BİRİKTİRİCİ (ACCUMULATOR)

Saatin her yükselen kenarında bir adet "M" bitlik (parametrik) ikiye tümleyen formatında sayı alan ve bu sayıyı önceki çevrimlerde gelen sayıların üstüne toplayan bir biriktirici (accumulator) modül gerçekleştirin. Biriktirici, toplama işleminin sonucunu parametrik olarak ("N" bit) tutacaktır. N parametresinin M parametresinden büyük olacağını varsayabilirsiniz. Toplamı sıfırlamak için modülün bir *reset* girişi olacaktır. Toplama işlemini Davranışsal Modelleme ile gerçekleştirebilirsiniz.

- **[Gerçekleştirme]** "clk" ismindeki saat sinyalinin her **yükselen kenarında** "sayi" isimli "M" bitlik girişten alınan ikiye tümleyen formatındaki sayıyı mevcut "N" bitlik toplama ekleyen modüle "Biriktirici" ismini verin. Toplamı, istenildiğinde sıfırlamak için kullanılacak olan girişe "rst" ismini verin. Toplamın o çevrimdeki değerini gösteren "N" bitlik çıkışa "toplam" ismini verin. Bu özelliklerdeki modülü Verilog ile davranışsal modelleme kullanarak gerçekleştirin.
- **[Simülasyon]** Gerçekleştirmiş olduğunuz "Biriktirici" modülünün istenildiği gibi çalıştığından emin olmak için, bu modülü test eden "tb_Biriktirici" adlı testbench kodunu yazın. Vivado yazılımını kullanarak modülün simülasyonunu yapıp, karedalga (Waveform) görünümünden modülün doğru çalıştığını kontrol edin.
- **[FPGA Kartı ile Deneme]** Simülasyon yaparak doğru çalıştığından emin olduğunuz "Biriktirici" modülünü kullanarak aşağıda belirtilen maddeleri sağlayan "FPGA_Biriktirici" isimli bir uygulama yazınız. Uygulamanızda,
  - 1 bitlik "clk" ve "rst" girişleri bulunmalıdır.
  - Biriktirici objesinin "M" parametresini 4, "N" parametresini 10 olarak belirleyin.
  - 10 bitlik "toplam" çıkışı bulunmalıdır. Toplam çıkışını FPGA kartınızın en sağdaki 10 LED'ini kullanarak ikilik tabanda gösterilecek şekilde ayarlayınız.
  - 4 bitlik "sayi" girişi bulunmalıdır. Sayi girişini FPGA kartınızın en sağdaki 4 Switch'ini kullanarak belirlenecek şekilde ayarlayınız.
  - "rst" girişini FPGA kartınızdaki orta buton ile belirlenecek şekilde ayarlayınız.
  - "clk" girişine bir saat bağlayınız.

  Uygulamanız için bir Bitstream üretin. FPGA'in Switch'leri yardımıyla istediğiniz değerleri verin, LED'de yanan değerin Switch'ler ile vermiş olduğunuz değerlerin toplamına karşılık geldiğini gözlemleyin.