**Laboratuvar Çalışması 0x1**

Birleşimli Mantık Devreleri

Bu labın amacı

* Donanım tanımlama dilleri ile *çözücü* (decoder) tasarlanması,
* Sentezleyici araçları kullanarak, tanımlanan devreleri FPGA için sentezlemek,
* Structural tasarım prensiplerini kavramak ve uygulamak,
* Simülasyon araçları kullanarak devre çalışmasını doğrulamak.

## Uyarılar

1. Lab'a başlamadan önce ders kadrosu tarafından verilen **ModelSim ve Quartus Prime kullanımı** adlı eğitim dokümanını 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 adlarını her zaman küçük harflerle **labX\_gY\_pZ.uzantı** olarak kaydedin.
   1. X yerine lab numarasını, Y yerine grup numaranızı, Z yerine problem numarasını ekleyin.
   2. Mesela lab3, SystemVerilog, Problem 1 ve grup numaranız 18 ise, **lab3\_g18\_p1.sv** olarak kaydedin.
   3. Devre adını da **lab3\_g18\_p1** olarak belirleyin.
   4. Testbench isimlerinin başına **tb\_** önekini koyun ve aksi belirtilmedikçe girişler arası bekleme süreleri için 10 ns kullanın.
4. Rapor isteniyorsa, 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.
6. ModelSim dalga şemasını File→Export→Image.. ile PNG olarak çıkarabilirsiniz. Veya tüm ModelSim projesinin ekran görüntüsü alabilirsiniz.
7. <https://github.com/fcayci/sv-digital-design> adresinde örnek HDL kodları mevcuttur.

## Problemler

### Problem 1 - Basit bir devre tasarımı ve simülasyonu

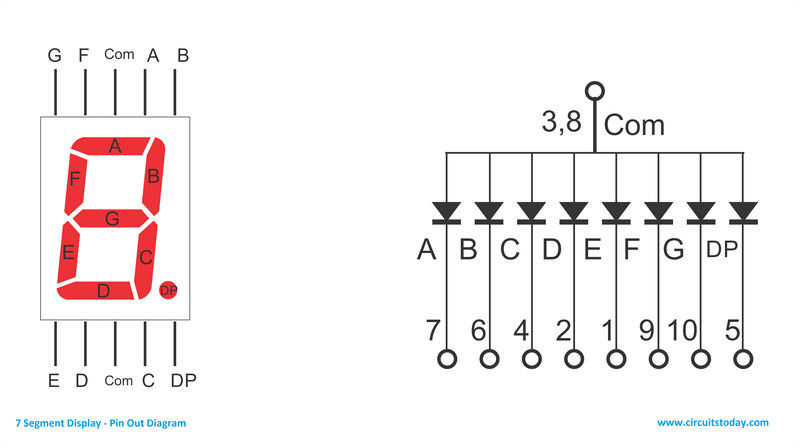
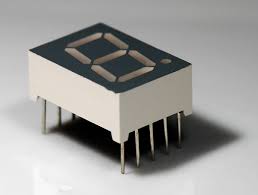
**Denklem 1**

1. Denklem 1 deki Boolean denklemini HDL ile tasarlayın. Örnek bir devre eklerde verilmiştir.
2. Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını gözlemleyin. Örnek bir testbench eklerde verilmiştir.
3. Devreyi Quartus’da sentezleyerek devrenin ne kadar yer kapladığını (resource utilization report), sentezlenen RTL ve eşleştirme ardı devre şemalarını ekleyerek **yorumlayınız**.

### Problem 2 – Toplayıcı tasarımı

* Bu problemde structural tasarım yaparak 4 bitlik bir ripple carry adder tasarlayacaksınız. Bu tasarımların çoğu yukarıda verilen adreste mevcuttur. [2]
* Genel olarak burada modüllerin birbirine nasıl bağlandığı ve bir modülün birkaç kez nasıl kullanıldığı kısımları önem arz etmektedir. (half\_adder.sv, full\_adder.sv, full\_adder\_str.sv, adder.sv, adder\_str.sv) modüllerini inceleyin.
* Sizin kullanacağınız modüller half\_adder ve \_str modülleri olacaktır. Diğerleri alternatif olarak nasıl olabilir göstermek için verilmiştir.
* adder\_str modülü generate bloğunun nasıl kullanıldığını göstermektedir. İsterseniz generate bloğunu kullanmayıp, her bir full adderı tek tek yazabilirsiniz. Eğer bağlantılarınız aynı ve doğru ise ikisi de aynı şekilde sentezlenecektir.

1. Half-adder tasarlayın
2. A da tasarladığınız half-adderları modül olarak kullanarak full-adder tasarlayın
3. B de tasarladığınız full-adderları modül olarak kullanarak 4-bitlik ripple carry adder tasarlayın.
4. Testbench oluşturarak, devrenin bütün girişlere karşı nasıl davrandığını ve sonuçların doğruluğunu gözlemleyin. (tb\_adder\_str kullanabilirsiniz. Doğrulama işlemini nasıl otomize edeceğinizi gösteren bir örnek)
5. Devreyi Quartus’da sentezleyerek devrenin ne kadar yer kapladığını, sentezlenen RTL ve eşleştirme ardı devre şemalarını ekleyerek **yorumlayınız**.



**Şekil 1** - 7 bölmeli ekran (7BE) (7-segment display) gösterilmektedir.[1] Her bir **ABCDEFG** bölmesi için ortak bir anode, ayrı bir cathode bulunmaktadır. Başka bir değişle, ortak pine **lojik 1**, **ABCDEFG** girişlerine ise **lojik 0** gönderildiğinde istenilen bölme yanacaktır.

### Problem 3 - Çözücü tasarımı

1. Şekil 1’deki 7BE elemanının bölmeleri Tablo 1 de verilen gösterimlerde yakmak için 4 girişli 7 çıkışlı bir *çözücü* (decoder) devresinin doğruluk tablosunu oluşturunuz ve boolean expressionlarını çıkarınız. İsterseniz K-Map ile sadeleştirip öyle devam edebilirsiniz.
2. Bu devreyi HDL ile tasarlayın. (Her bir 7BE çıkışı için bir assign olacaktır.)
3. Testbench oluşturarak, 16 farklı giriş kombinasyonuna göre çıkış dalga şeklini gözlemleyiniz. Tabloda verilen girişlerde doğru çıkışların olduğunu doğrulayınız.
4. Devreyi Quartus’da sentezleyerek devrenin ne kadar yer kapladığını, sentezlenen RTL ve eşleştirme ardı devre şemalarını ekleyerek **yorumlayınız**.

|  |  |
| --- | --- |
| Girişler (**x3x2x1x0**) | 7BE gösterimi |
| 0000 | **-** |
| 0010 | **E** |
| 0011 | **L** |
| 0111 | **C** |
| 1001 | **2** |
| 1011 | **3** |
| 1111 | **5** |

Tablo 1

## IV. Referanslar

1. <https://en.wikipedia.org/wiki/Seven-segment_display>
2. <https://github.com/fcayci/sv-digital-design>

## V. Ekler

### EK A - Örnek HDL kodları

|  |
| --- |
| /\* lab1\_g0\_p0.sv  \*  \* Hazırlayanlar:  \* Furkan Çaycı  \*  \* Notlar:  \* Y = NOT A and B denkleminin gerçeklemesi  \*  \*/  module lab1\_g0\_p0 (  input logic a, b,  output logic y  );  assign y = ~a & b;  endmodule |

|  |
| --- |
| /\* tb\_lab1\_g0\_p0.sv  \*  \* Hazırlayanlar:  \* Furkan Çaycı  \*  \* Notlar:  \* 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\_lab1\_g0\_p0 ();  // 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.  lab1\_g0\_p0 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. Important when to stop.  end  endmodule |