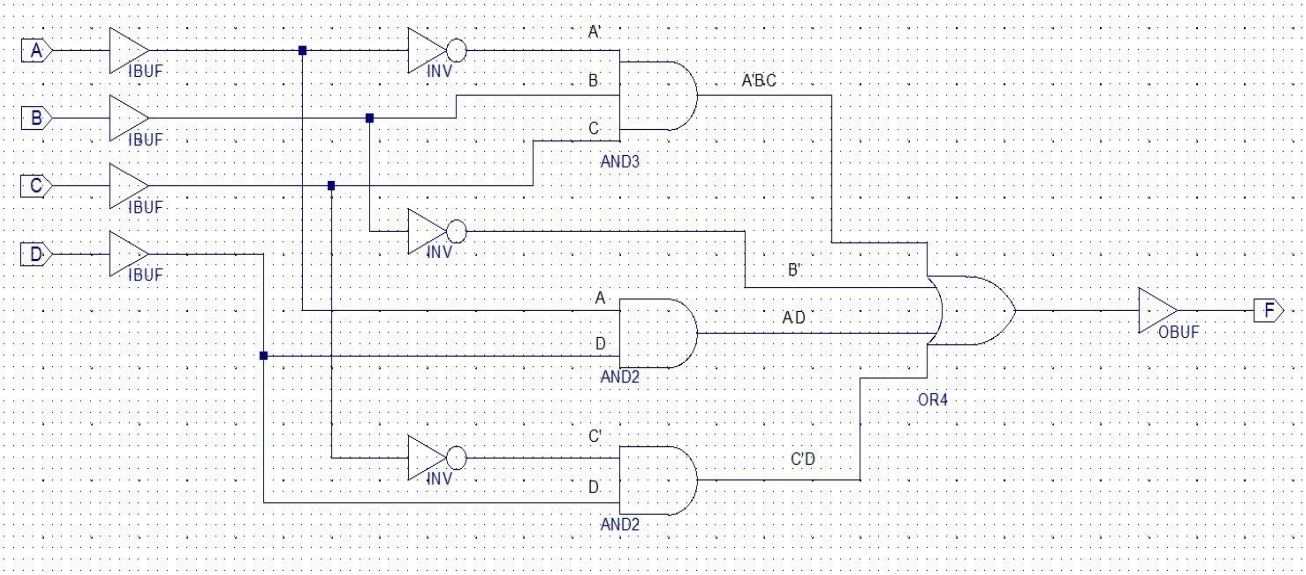
Verilog ile Kapı Seviyesinde Tasarım

Verilog, donanım tasarlamak için geliştirişmiş bir programlama dilidir. Verilog benzeri dillere Hardware Description Language (Donanım Tarifleme Dili) denir. Verilog ile tariflenen bir donanım Computer-aided Design (CAD) araçları kullanılarak istenilen platform için donanım gerçeklenmesi yapılır. CAD araçlarını belli bir platform için (FPGA gibi) donanım sentezleyen derleyici (compiler) gibi düşünebilirsiniz. Xilinx ISE uygulamasında Verilog kodu alınarak sırasıyla "Synthesis", "Implementation" ve "Generate Programming File" gibi adımlar kaynak kodunun otomatik olarak uygun formata dönüştürülerek, seçilen FPGA'a yüklenmesine olanak sağlar.

Bu tutorial'da Verilog diline basit bir başlangıç yapacağız. Bil264L/ELE263L dersleri kapsamında Verilog ile Kapı seviyesinde ve Davranışsal modelleme yöntemlerini inceleyeceğiz. Dökümanın devamında kapı seviseyinde tasarım küçük bir örnek verilerek anlatılmıştır.

Birinci lab dersinde şematik tasarımını yapmış olduğunuz f fonksiyonu ve şematik gösterimi aşağıda verilmiştir.

f (A,B,C,D) = A'BC + B' + AD + C'D = F

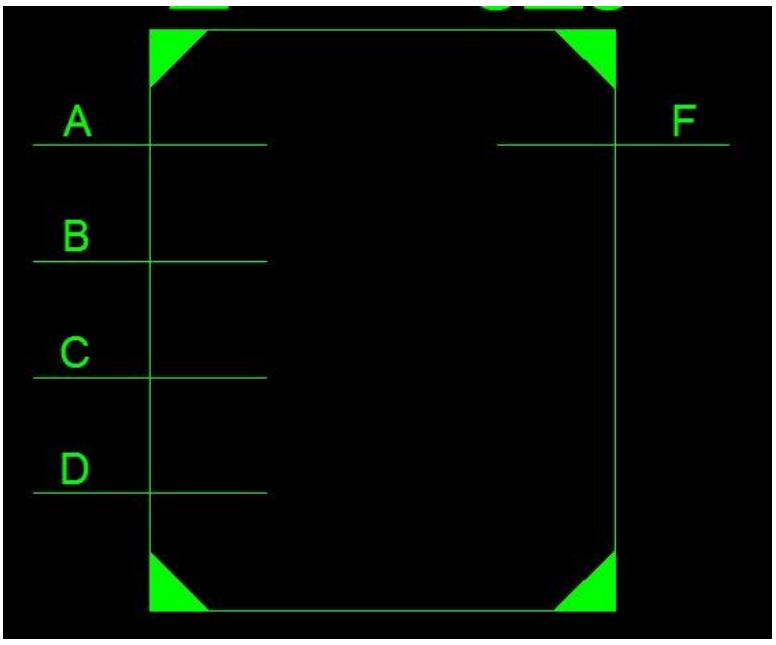


Aynı donanımı tarifleyen kapı seviyesinde Verilog kodu aşağıdaki gibi yazılabilir:

```
module lab1_verilog_gate(
                input A, B, C, D,
                output F
        );
        wire Ap, Bp, Cp;
        wire ApBC, AD, CpD;

        not n1(Ap, A);
        and g1(ApBC, Ap, B, C);

        not n2(Bp, B);

        and g2(AD, A, D);

        not n3(Cp, C);
        and g3(CpD, Cp, D);

        or g4(F, ApBC, Bp, AD, CpD);
endmodule
```

Verilog kodu yazabilmek için projenize yeni bir Verilog modülü dosyası eklemeiz gerekir. Bunun için prejenizin “Hierarchy” bölümüne sağ tıklayıp “New Source” a tıklayıp, daha sonra Source Type’ı “Verilog Module” olarak seçip yeni bir dosya oluşturabilirsiniz.

Verilog ile donanım tarifleme modüler olarak yapılır. Modüller giriş/çıkış portları olan kapalı kutular gibi düşünülebilir. Örneğin; f fonksiyonu için oluşturacağımız modül aşağıdaki gibi gösterilebilir.



A, B, C ve D giriş (input), F ise çıkış (output) portlarıdır.

Bir modülün içinde daha önceden tasarlanmış olan başka bir modül kullanılabilir. Modüler tasarım sayesinde belli bir işi yapan bir donanımı bir kere tasarlayıp, daha sonra farklı devreler için tekrar tekrar kullanabiliriz.

Verilog ile yapmak istediğimiz modülün giriş ve çıkışlarını belirtirken aşağıdaki syntax kullanılır:

```
module modul_ismi(
                    input giris1,
                    input giris2,
                    output cikis1
          );

endmodule
```

Burada "module" ve "endmodule" keyword'lerdir ve modülün başlangıcında ve bitişinde sırasıyla yazılmalıdır. "modul_ismi" olarak istediğimiz adı verebiliriz (keyword'ler hariç) ve modül isminden sonra parantez açarak modülün giriş ve çıkışlarını input/output keyword'lerini kullanarak belirtiyoruz. Giriş/çıkışlar istenilen sayıda olabilir ve istenilen isim kullanılabilir.

Örnekte dört girişli ve tek çıkışlı modül için şu kod yazılmıştır:

```
module lab1_verilog_gate(
                    input A, B, C, D,
                    output F
          );
```

Bir kere "input" yazdıktan sonra virgülle ayırarak tüm input portlarını da belirtmek mümkündür.

Bir başka keyword de "wire" dır. Wire ile kendisine daha sonra atama yapacağımız değişkenleri belirtiyoruz. Örnek kodda kullanıldığı yer şu şekildedir:

```
wire Ap, Bp, Cp;
wire ApBC, AD, CpD;
```

Bunlar f fonksiyonunu gerçeklerken kullanılan "and, or, not" gibi kapıların ara çıktılarıdır. Wire türündeki veriler istenildiği gibi isimlendirilebilir. Verilen örnekteki isimlendirmede büyük harfler kullanılan girdileri, küçük "p" harfi ise "prime" yani o girdinin değilinin alındığını göstermektedir. Örneğin; Bp wire'ına B'nin değili atanacaktır.

Verilog'da primitive gate'ler (kapı) aşağıdaki şekilde tanımlanır.

```
        and U2(out1, in1, in2, in3, in4);
```

Yukarıdaki bir "and" kapısı olup "out1" adında çıkışı, "in1", "in2", "in3" ve "in4" adlarında da girişleri vardır. U2 ise bu girişlere ve çıkışa sahip olan kapıya bizim verdiğimiz bir isimdir.

F fonksiyonumuza dönecek olursak:

```
not n1(Ap, A);
and g1(ApBC, Ap, B, C);

not n2(Bp, B);

and g2(AD, A, D);

not n3(Cp, C);
and g3(CpD, Cp, D);

or g4(F, ApBC, Bp, AD, CpD);
```

Şematikte verilen her bir “and”, “or” ve “not” gibi elemanlar Verilog’daki gate primitive’leri kullanarak yukarıdaki gibi gerçeklenmiştir. Giriş ve çıkışları dikkatlice incelerseniz kodun şematikte verilen devreyi gerçeklediğini görebilirsiniz. Ayrıca bu devreyi geçen haftanın testbench kodunu kullanarak da doğrulayabilirsiniz.