Electronic Circuits

Week 9: Classes

Fenerbahçe University

Professor & TAs

Prof: Dr. Vecdi Emre Levent

Office: 311

Email: emre.levent@fbu.edu.tr

TA: Arş. Gör. Uğur Özbalkan

Office: 311

Email: ugur.ozbalkan@fbu.edu.tr

Classes

System Verilog

- Verilog dilinin üzerine eklemeler yapılarak oluşturulmuş bir dildir.
- Genellikle ileri seviye doğrulama işlemleri için tercih edilir.
- Doğrulama için Synopsys VCS ve Menthor Graphics QuestaSim gibi araçlar kullanılabilir.

Classes

System Verilog

- Ancak piyasada bulunan Synopsys Synplify ve Mentor Precision gibi sentez araçları System Verilog ile sentez yapılmasına izin vermektedir. Xilinx Vivado aracı da system verilog ile sentezleme yapabilmektedir. Ancak diğer araçlara göre System Verilog desteği daha kısıtlıdır.

Classes

System Verilog

- System Verilog kullanımının Verilog diline göre çeşitli avantajları aşağıdaki gibi sıralanabilir.

- Daha kompakt bir kod
- Daha yüksek seviyeden bir yaklaşım ile tasarım
- Barındırdığı yapılar ile tasarımı büyütmek daha kolay

- System verilog ile tasarım dosyalarının uzantısı ”sv” olmaktadır.

Classes

System Verilog

- Nesneye yönelimli geliştirme konsepti bir çok programlama dilinde var olan bir geliştirme konseptidir.
- Bu geliştirme konsepti ile geliştirilen uygulamalar yeni ihtiyaçlar doğduğunda çok daha hızlı adapte edilebilir, esnek ve yeniden kullanımı yüksek kodlar ortaya çıkmaktadır.
- System Verilog nesneye yönelimli programlama konseptlerini desteklemektedir.

Classes

System Verilog

- Sınıflar, nesneye yönelimli programlama dillerinde ve System Verilog gibi özellikle gelişmiş test durumları yazmak için kullanılan HDL dillerinde daha esnek programlar geliştirebilmek için kullanılan bir konsepttir.

Classes

System Verilog

- Bir arada sıklıkla kullanılan ve/veya aynı görev için kullanılan değişken ve fonksiyonları gruplar. Bu gruptan yani sınıftan istendiğinde objeler türetilerek o fonksiyon ve değişkenlerden bir kopya yaratılmış olunur.

Classes

System Verilog

- Geliştirme esnekliği; sadece sınıfın barındırdığı fonksiyon veya değişkenlerde bir değişiklik yapıldığında, o değişiklik sınıftan türetilmiş tüm objelere anında aktarılacağı için kodu daha modüler hale getirmesinden gelmektedir.

Classes

System Verilog

- Sınıflarda veri elemanları (wire, reg, bit, int) ve fonksiyonlar (task, function) ‘lar bulunmaktadır. Sınıflardan istendiği zaman dinamik olarak (Çalışma zamanında) objeler türetilip silinebilir. Aşağıda iki adet system verilog syntax’inde sınıf örneği verilmektedir.

Classes

System Verilog

```
module testModulu;

    class testClass;
        int sayi;
    endclass

endmodule
```

```
module testModulu;

    class goruntu;
        bit [31:0] count;
        bit [3:0] parity;
        int number;

        task setFnc (input in i);
            number = i;
        endtask

        function int getFnc();
            return number;
        endfunction

    endclass

endmodule
```

Classes

System Verilog

Sınıflardan bir obje türetmek için türetilecek olan sınıfın ismi ve yanına obje ismi yazılır.

```
module testModulu;

    class testClass;
        int sayi;
    endclass

    testClass obj1;

endmodule
```

Classes

System Verilog

- Sınıftan bir obje türetildiği zaman henüz aslında RAM'de sınıftaki değişkenler için yer açılmaz. Yani objenin DRAM'de tutulduğu bir adres yoktur.
- Bu yüzden NULL değerini taşımaktadır.
- Objenin bellek tahsisi için "new" operatörü ile atama yapılması gerekmektedir.
- New ifadesi sonucunda, obj1 DRAM'de atanan adresi göstermektedir.

Classes

System Verilog

```
module testModulu;

    class testClass;
        int sayi;
    endclass

    testClass obj1;

    initial begin
        obj1 = new;
    end

    obj2 = new;

endmodule
```

Classes

System Verilog

- Bazı nesneye yönelimli programlama dillerinde “Garbage Collection” denen bir mekanizma bulunmaktadır.
- Bu mekanizma, artık kullanılmayan objeleri otomatik olarak DRAM’den kaldırılmasını sağlamaktadır.
- Bu özellik DRAM’in şişmesini otomatik olarak engellemekte, tasarımcının kullanılmayan değişkenleri kendisinin takip edip silmesi gereksinimini ortadan kaldırmaktadır.
- System Verilog’da da bu özellik mevcuttur.

Classes

System Verilog

- Yine bazı programlama dillerinde “Destructor” denen bir özel fonksiyon bulunur. Bu fonksiyon obje tam DRAM’den silineceği esnada otomatik olarak çağrılır.
- Obje silinirken otomatik olarak yapılması istenen bir işlem varsa bu fonksiyonda tanımlanabilir.
- Bu özellik System Verilog’da yoktur.

Classes

System Verilog

- Sınıf içindeki fonksiyon ve değişkenlere, sınıftan türetilmiş objenin adı'nın yanına nokta işareti koyularak erişilmektedir.
- Değişkenler ve fonksiyonlara erişim doğrudan (Public olduğu sürece) veya fonksiyonlar aracılığı ile dolaylı olabilir.

Classes

System Verilog

```
module testModulu;
    class goruntu;
        bit [31:0] count;
        bit [3:0] parity;
        int number;

        task setFnc (input in i);
            number = i;
        endtask

        function int getFnc();
            return number;
        endfunction

    endclass

    goruntu obj1 = new;
    goruntu obj2 = new;

    initial begin
        obj1.number = 1;
        obj2.setFnc(3);

        $display("Obj1 sayi %d", obj1.number);
        $display("Obj2 sayi %d", obj2.getFnc());

    end
endmodule
```

Classes

System Verilog

- Birçok programlama dilinde, sınıfın içindeki fonksiyonların gerçeklemeleri ayrı bir dosyada tutulabilmektedir.
- Bunun nedeni, kodun gerçeklemelerini farklı bir tasarımcıya açmadan sadece fonksiyonların girişlerinin ne olduğu ve ne döndürdüğü bilgisi vererek bilgi gizlemektir.
- Ayrıca kodun okunabilirliğini arttırmak için de kullanılabilir.
- Systemverilog'da bu özelliği desteklemektedir.
- Bunun için bir sınıfın içindeki fonksiyonun prototipini yazmak gerekmektedir. "extern" syntax'i ile yapılabilmektedir.

Classes

System Verilog

```
module testModulu;

    class goruntu;
        int number;

        extern int denemeFonksiyonu();

    endclass

    function int testModulu::denemeFonksiyonu();
        return number;
    endfunction

endmodule
```

Classes

System Verilog

- Programlama dillerinde sınıf konseptlerinde “Constructor” denen özel bir fonksiyon bulunur. Bu mekanizma bir sınıftan obje türetileceğinde, obje türetilirken başlangıçta yapılması istenen bazı işlemler varsa, o işlemleri gerçekleştirmek için kullanılabilir.

Classes

System Verilog

- Aşağıdaki kod parçacığında goruntu sınıfından obj1 isminde bir obje türetilmiştir. Bu objenin aslında görünmeyen bir constructor'u vardır. Başlangıçta number değişkenine 0 atamasını yapmıştır.

```
module testModulu;

    class goruntu;
        int number;
    endclass

    initial begin
        goruntu obj1 = new;
    end

endmodule
```

Classes

System Verilog

- Aşağıdaki kod parçacığında ise sınıfın içerisine new isminde bir fonksiyon yazıldığı görülmektedir. Bu fonksiyon constructor'dur. Constructor'da number değişkenne başlangıç ataması yapılmaktadır. Constructor bir argüman almamaktadır.

```
module testModulu;

    class goruntu;
        int number;

        function new();
            number = 3;
        endfunction

    endclass

    initial begin
        goruntu obj1 = new;
    end

endmodule
```

Classes

System Verilog

- Constructor istendiğinde argüman alabilmektedir.
- Bunun için constructor foksiyonunu argüman alacak şekilde düzenleyip, sınıftan obje türetilirken kullanılan new ifadesine argümanları beslemek gerekmektedir.

Classes

System Verilog

```
module testModulu;

    class goruntu;
        int number;

        function new(input int arg1, input int arg2);
            number = arg1 * arg2;
        endfunction

    endclass

    initial begin
        goruntu obj1 = new(3, 5);
    end

endmodule
```