Electronic Circuits

Week 12: Variables

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

Variable Types

String, karakterleri bir arada tutan bir veri türüdür. String değişkenin uzunluğu dinamiktir. String'lerin sabit bir karakter genişliği yoktur.

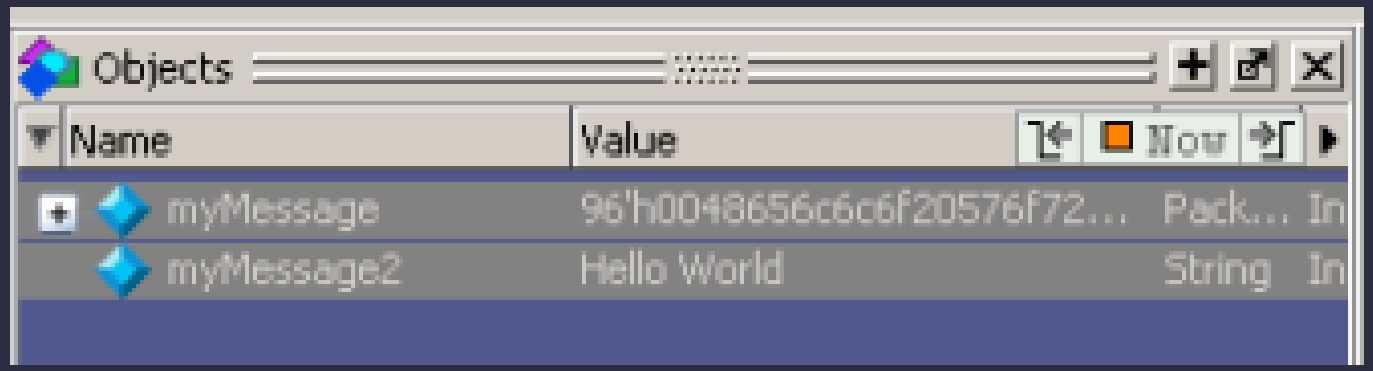
Sytanx'ı:

string değişkenAdi

Variable Types

```
timeunit 1ns; timeprecision 1ns;

module top;

bit [8*12 -1:0] myMessage  = "Hello World";
string          myMessage2 = "Hello World";

endmodule: top
//`end_keywords
```

Variable Types



String Gösterimi

Variable Types

String bir ifadenin istenen indeksteki karakterine erişilebilir.

```
module tb;

  string test = "Hello!";

  initial begin

    $display ("%s", test);

    foreach (test[i]) begin
      $display ("%s", test[i]);
    end
  end
endmodule
```

Variable Types

```
# Hello!
# H
# e
# l
# l
# o
# !
```

String Konsol Çıktıları

Variable Types

String'in bir çok hazır fonksiyonu bulunmaktadır. Bu fonksiyonlar özetlenmiştir.

Kullanım	Tanım	Açıklama
str.len()	function int len()	String'in boyutunu döndürür
str.putc()	function void putc (int i, byte c);	Verilen index'teki karakterin üzerine, verilen karakteri kopyalar
str.getc()	function byte getc (int i);	Verilen index'teki karakteri geri döndürür
str.tolower()	function string tolower();	String'in tüm karakterlerini küçük olarak geri döndürür
str.compare(s)	function int compare (string s);	İki string'i aynı olup olmadığını case sensitive karşılaştırır.
str.icompare(s)	function int icompare (string s);	İki string'i aynı olup olmadığını case sensitive olmadan karşılaştırır.
str.substr (i, j)	function string substr (int i, int j);	Verilen indeks aralıklarından alt string'i geri döndürür

Variable Types

string fonksiyonları testi

```
module tb;
    string str1 = "Hello World!";
    string str2 = "Hello World!";

    initial begin

        $display ("str.len() = %0d", str1.len());
        str1.putc (3,"X");
        $display ("%s", str1);
        $display ("%s (%0d)", str1.getc(2), str1.getc(2));
        $display ("%s", str1.tolower());
        $display ("Compare %0d", str1.compare(str2));
        str1 = "hello world!";
        $display ("Compare %0d", str1.compare(str2));
        $display ("ICompare %0d", str1.icompare(str2));
        $display ("Substr str1 = %s", str1.substr (4,8));

    end
endmodule
```

Variable Types

```
# str.len() = 12
# HelXo World!
# l (108)
# helxo world!
# Compare -20
# Compare 32
# ICompare 0
# Substr str1 = o wor
```

String Fonksiyonları

Variable Types

String Dönüşüm Fonksiyonları

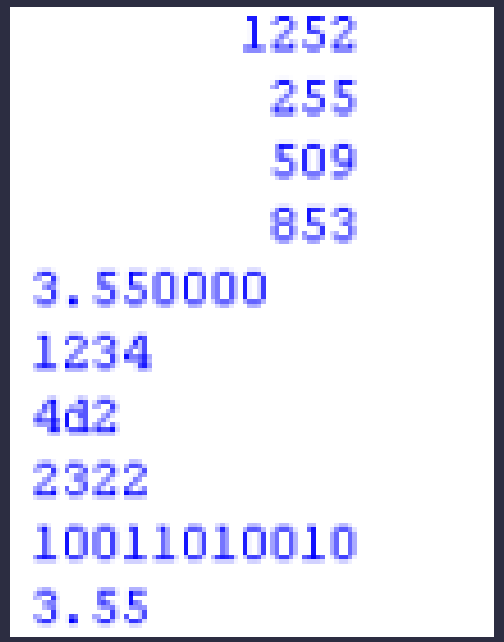
Kullanım	Tanım	Açıklama
str.atoi()	function integer atoi();	ASCII karakterin karşılığı sayıyı döndürür
str.atohex()	function integer atohex();	String'de hex sayılar olarak yorumlar ve geriye tam sayı döndürür
str.atooct()	function integer atooct();	String'de octal sayılar olarak yorumlar ve geriye tam sayı döndürür
str.atobin()	function integer atobin();	String'de binary sayılar olarak yorumlar ve geriye tam sayı döndürür
str.atoreal()	function real atoreal();	String'de real sayılar olarak yorumlar ve geriye tam sayı döndürür
str.itoa(i)	function void itoa (integer i);	Verilen tam sayının karşılığı ASCII karakteri yerleştirir
str.hextoa(i)	function void hextoa (integer i);	Verilen tam sayının hex halini yerleştirir
str.octtoa(i)	function void octtoa (integer i);	Verilen tam sayının octal halini yerleştirir
str.bintoa(i)	function void bintoa (integer i);	Verilen tam sayının binary halini yerleştirir
str.realtoa(r)	function void realtoa (real r);	Verilen tam sayının real sayı halini yerleştirir

Variable Types

String dönüşüm örnekleri

```
module tb;

    string str1 = "1252";
    int temp = 0;
    real temp2 = 0;

    initial begin

        temp = str1.atoi();
        $display("%d", temp);

        str1 = "FF";
        temp = str1.atohex();
        $display("%d",temp);

        str1 = "775";
        temp = str1.atooct();
        $display("%d",temp);

        str1 = "1101010101";
        temp = str1.atobin();
        $display("%d", temp);

        str1 = "3.55";
        temp2 = str1.atoreal();
        $display("%f", temp2);

        // Int to dönüşümler

        str1.itoa(1234);
        $display("%s", str1);

        str1.hextoa(1234);
        $display("%s", str1);

        str1.octtoa(1234);
        $display("%s", str1);

        str1.bintoa(1234);
        $display("%s", str1);

        str1.realtoa(3.55);
        $display("%s", str1);
    end
endmodule
```

Variable Types



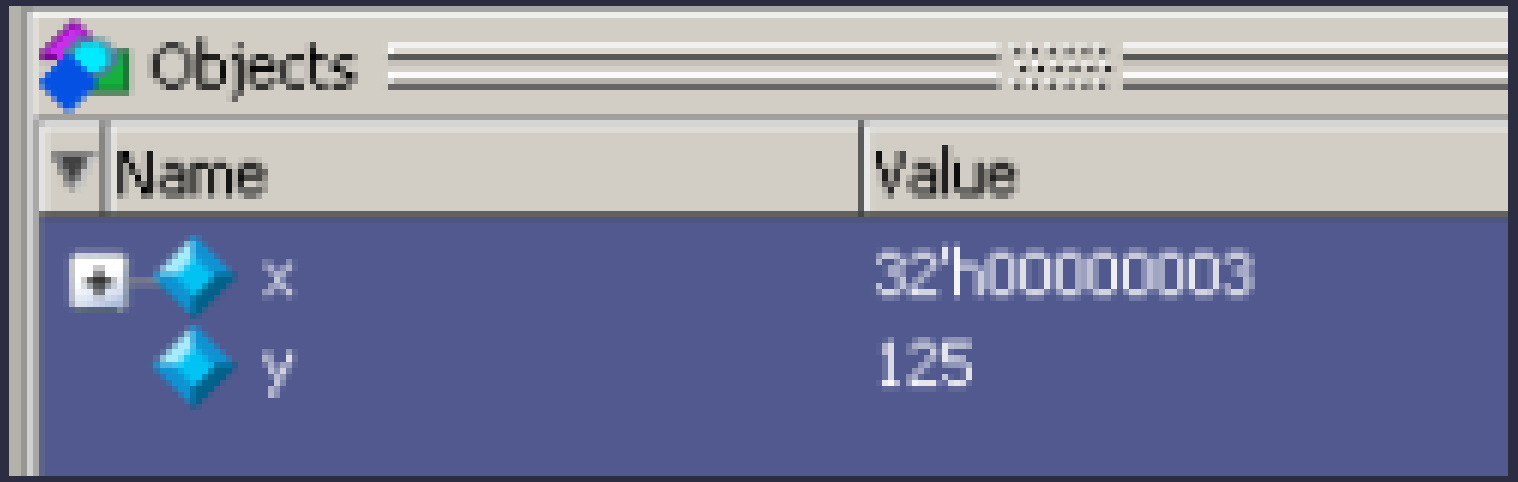
String Dönüşüm Örneği Konsol Çıktıları

Variable Types

Ayrıca real ve integer arasında da dönüşüm yapılabilmektedir.

```
module tb;

    int x;
    real y;
    initial begin
        x = $rtoi ( 3.55 );
        y = $itor ( 125 );
    end

endmodule
```

Variable Types



Real-Integer Dönüşüm Sonucu

Variable Types

Enumeration

Enumeration, isimlendirilmiş sayı değerleri tanımlar. Kodun okunabilirliğini arttırmak için tercih edilirler.

```
enum          {RED, YELLOW, GREEN}          light_1;
// int type; RED = 0, YELLOW = 1, GREEN = 2
enum bit[1:0] {RED, YELLOW, GREEN}          light_2;
// bit type; RED = 0, YELLOW = 1, GREEN = 2
```

Tasarımcı isimlendirmelerin başlangıç değerlerini güncelleyebilir.

```
enum          {RED=3, YELLOW, GREEN}       light_3;          // RED = 3, YELLOW =
4, GREEN = 5
enum          {RED = 4, YELLOW = 9, GREEN} light_4;          // RED = 4, YELLOW =
9, GREEN = 10
enum          {RED = 2, YELLOW, GREEN = 3} light_5;          // Hata : YELLOW ve
GREEN aynı değere atandı
enum bit[0:0] {RED, YELLOW, GREEN} light_6;                  // Hata: en az 2 bit
gerekir
```

Variable Types

Enumeration’ların sahip olduğu fonksiyonlar

Kullanım	Tanım	Açıklama
first()	function enum first();	İlk elemanın değerini döndürür
last()	function enum last();	Son elemanın değerini döndürür
next()	function enum next (int unsigned N = 1);	Çağrılan enum'un bir sonraki enum'un değerini döndürür
prev()	function enum prev (int unsigned N = 1);	Çağrılan enum'un bir önceki enum'un değerini döndürür
num()	function int num();	Eleman sayısını döndürür
name()	function string name();	Metin gösterimini döndürür

Variable Types

```
module tb;

    typedef enum {TRUE, FALSE} e_true_false;

    initial begin
        e_true_false  answer;

        answer = TRUE;

        $display ("answer = %s", answer.name);
    end
endmodule
```

```
# answer = TRUE
```

Variable Types

Enumeration'ların sahip olduğu fonksiyonlar

```
module tb;
  // GREEN = 0, YELLOW = 1, RED = 2, BLUE = 3
  typedef enum {GREEN, YELLOW, RED, BLUE}
color_set_1;

  // MAGENTA = 2, VIOLET = 7, PURPLE = 8,
PINK = 9
  typedef enum {MAGENTA=2, VIOLET=7, PURPLE,
PINK} color_set_2;

  // BLACK0 = 0, BLACK1 = 1, BLACK2 = 2,
BLACK3 = 3
  typedef enum {BLACK[4]} color_set_3;
  // RED0 = 5, RED1 = 6, RED2 = 7
  typedef enum {RED[3] = 5} color_set_4;
  // YELLOW3 = 0, YELLOW4 = 1, YELLOW5 = 2
  typedef enum {YELLOW[3:5]} color_set_5;
  // WHITE3 = 4, WHITE4 = 5, WHITE5 = 6
  typedef enum {WHITE[3:5] = 4} color_set_6;
```

```
initial begin
    color_set_1 color1;
    color_set_2 color2;
    color_set_3 color3;
    color_set_4 color4;
    color_set_5 color5;
    color_set_6 color6;

    color1 = YELLOW; $display ("color1 = %0d, name = %s",
color1, color1.name());
    color2 = PURPLE; $display ("color2 = %0d, name = %s",
color2, color2.name());
    color3 = BLACK3; $display ("color3 = %0d, name = %s",
color3, color3.name());
    color4 = RED1;   $display ("color4 = %0d, name = %s",
color4, color4.name());
    color5 = YELLOW3;$display ("color5 = %0d, name = %s",
color5, color5.name());
    color6 = WHITE4; $display ("color6 = %0d, name = %s",
color6, color6.name());

  end
endmodule
```

Variable Types

```
# color1 = 1, name = YELLOW
# color2 = 8, name = PURPLE
# color3 = 3, name = BLACK3
# color4 = 6, name = RED1
# color5 = 0, name = YELLOW3
# color6 = 5, name = WHITE4
```

Enum Konsol Çıktıları

Variable Types

Enum Fonksiyon Testleri

```
typedef enum {GREEN, YELLOW, RED, BLUE} colors;

module tb;

    initial begin
      colors color;

      color = YELLOW;

      $display ("color.first() = %0d", color.first());
      $display ("color.last()  = %0d", color.last());
      $display ("color.next()  = %0d", color.next());
      $display ("color.prev()  = %0d", color.prev());
      $display ("color.num()   = %0d", color.num());
      $display ("color.name()  = %s" , color.name());
    end

endmodule
```

Variable Types

color.first() = 0
color.last() = 3
color.next() = 2
color.prev() = 0
color.num() = 4
color.name() = YELLOW

Enum’lar bir tamsayı içerselerde, üzerilerine sonradan bir tamsayı ataması yapılamaz.

Variable Types

Diziler

```
module tb;

    int   myFIFO  [0:7];
    int   urFIFO  [8];

    int   myArray  [2][3];

    initial begin
      myFIFO[5] = 32'hABCDEF01;
      myArray [1][1] = 7;

      foreach (myFIFO[i])
        $display ("myFIFO[%0d] = 0x%0h", i, myFIFO[i]);

      foreach (myArray[i])
        foreach (myArray[i][j])
          $display ("myArray[%0d][%0d] = %0d", i, j, myArray[i][j]);

    end
endmodule
```

Variable Types

```
myFIFO[0] = 0x0
myFIFO[1] = 0x0
myFIFO[2] = 0x0
myFIFO[3] = 0x0
myFIFO[4] = 0x0
myFIFO[5] = 0xabcdef01
myFIFO[6] = 0x0
myFIFO[7] = 0x0
myArray[0][0] = 0
myArray[0][1] = 0
myArray[0][2] = 0
myArray[1][0] = 0
myArray[1][1] = 7
myArray[1][2] = 0
```

Dizi Konsol Çıktısı

Variable Types

SystemVerilog farklı türde dizi tiplerini desteklemektedir. Bunlar:

- Statik
- Dinamik
- Associative
- Queue

olarak listelenebilir.

Variable Types

Statik diziler, boyutları derleme zamanında belli olan dizi türleridir.

```
Module tb;
    bit [7:0]   m_data; //1d packed dizi

    initial begin
        m_data = 8'hA2;

        for (int i = 0; i < $size(m_data); i++) begin
            $display ("m_data[%0d] = %b", i, m_data[i]);
        end
    end
endmodule
```

Variable Types

Statik diziler, boyutları derleme zamanında belli olan dizi türleridir.

```
# m_data[0] = 0
# m_data[1] = 1
# m_data[2] = 0
# m_data[3] = 0
# m_data[4] = 0
# m_data[5] = 1
# m_data[6] = 0
# m_data[7] = 1
```

Variable Types

Dinamik diziler, derlenme zamanında boyutunun belli olmadığı dizi türleridir.
Çalışma zamanında gereksinimlere göre boyutları değişebilmektedir.

```
int m_mem []; // Dinamik olarak boyutu değiştirilebilir.
```

```
module tb;
    int array [];

    initial begin
        array = new [5];

        array = '{31, 67, 10, 4, 99};

        foreach (array[i])
            $display ("array[%0d] = %0d", i, array[i]);
    end
endmodule
```

Variable Types

```
#  array[0] = 31
#  array[1] = 67
#  array[2] = 10
#  array[3] = 4
#  array[4] = 99
```

Dinamik Dizi Konsol Çıktıları

Variable Types

```
module tb;
    string  fruits [];

    initial begin
        fruits = new [3];

        fruits = '{"apple", "orange", "mango"};

        $display ("fruits.size() = %0d", fruits.size());

        fruits.delete();

        $display ("fruits.size() = %0d", fruits.size());
    end

endmodule
```

Dinamik Silme Konsol Çıktısı

```
# fruits.size() = 3
# fruits.size() = 0
```

Variable Types

Associative dizilerde, elemanlara bir anahtar aracılığı ile erişilir.

```
int m_data [int];   // Integer index
int m_name [string]; // String index

m_name ["Rachel"] = 30;
m_name ["Orange"] = 2;

m_data [32'h123] = 3333;
```

Variable Types

```
module tb;

    int     array1 [int];
    int     array2 [string];
    string  array3 [string];

    initial begin
        array1 = '{ 1 : 22, 6 : 34 };

        array2 = '{ "Ali" : 100, "Ahmet" : 60 };

        array3 = '{ "Test" : "3", "Deneme" : "XYZ" };

      $display ("array1 = %p", array1);
      $display ("array2 = %p", array2);
      $display ("array3 = %p", array3);
    end

endmodule
```

```
# array1 = '{1:22, 6:34 }
# array2 = '{"Ahmet":60, "Ali":100 }
# array3 = '{"Deneme":"XYZ", "Test":"3" }
```

Associative Array Çıktısı

Variable Types

Queue'lar push ve pop işlemleri ile çalışan bir veri saklama grubudur. FIFO yapısıdır. İlk yazılan ilk okunur. $ operatörü ile queue tanımı yapılır.

```
int m_queue [$];
m_queue.push_back(23);
int data = m_queue.pop_front();
```

Variable Types

```
module tb;
    string elemanlar[$] = {"Eleman1", "Eleman2", "Eleman3", "Eleman4"};

    initial begin
        $display ("Queue eleman sayisi=%0d   elemanlar=%p", elemanlar.size(), elemanlar);

        elemanlar.insert (1, "Eleman5");
        $display ("Eleman Sayisi=%0d elemanlar=%p", elemanlar.size(), elemanlar);

        elemanlar.delete (3);
        $display ("Eleman Sayisi=%0d elemanlar=%p", elemanlar.size(), elemanlar);

        // pop_front() - Öndeki elemanı alır
        $display ("Pop %s,    sayi=%0d elemanlar=%p", elemanlar.pop_front(), elemanlar.size(),
elemanlar);

        // push_front() - Öne eleman ekler
        elemanlar.push_front("Eleman6");
        $display ("Push, sayi=%0d elemanlar=%p", elemanlar.size(), elemanlar);

        // pop_back() - arkadan eleman alir
        $display ("Pop %s,   sayi=%0d elemanlar=%p", elemanlar.pop_back(), elemanlar.size(),
elemanlar);

        // push_back() - Arkaya eleman ekler
        elemanlar.push_back("Eleman7");
        $display ("Push,    sayi=%0d elemanlar=%p", elemanlar.size(), elemanlar);
    end
endmodule
```

Variable Types

```
# Queue eleman sayisi=4   elemanlar='{"Eleman1", "Eleman2", "Eleman3", "Eleman4"}
# Eleman Sayisi=5 elemanlar='{"Eleman1", "Eleman5", "Eleman2", "Eleman3", "Eleman4"}
# Eleman Sayisi=4 elemanlar='{"Eleman1", "Eleman5", "Eleman2", "Eleman4"}
# Pop Eleman1,    sayi=3 elemanlar='{"Eleman5", "Eleman2", "Eleman4"}
# Push, sayi=4 elemanlar='{"Eleman6", "Eleman5", "Eleman2", "Eleman4"}
# Pop Eleman4,   sayi=3 elemanlar='{"Eleman6", "Eleman5", "Eleman2"}
# Push,    sayi=4 elemanlar='{"Eleman6", "Eleman5", "Eleman2", "Eleman7"}
```

Queue Elamanları

Variable Types

Bir struct birden çok farklı türdeki elemanı gruplamak için kullanılan bir yapıdır.

Syntax'ı:

```
struct {
    int var1;
    bit var2;
    string var3;
} structAdi;
```

Variable Types

Bir struct birden çok farklı türdeki elemanı gruplamak için kullanılan bir yapıdır.

Syntax'ı:

```
module tb;

    typedef struct {
        int     coins;
        real    dollars;
    } test;

    test test1;
    test test2 [1:0] = '{'{2, 4.25}, '{7,1.5}};

    initial begin
        test1 = '{5, 19.75};
        test1 = '{coins:5, dollars:19.75};
        test1 = '{default:0};
        test1 = test'{int:1, dollars:2};
    end

endmodule
```

Variable Types

Unionlar ise aynı struct'lar gibidir ancak union'da tanımlanmış değişkenlerin tamamı aynı bellek bölgesini kullanır yani aynı anda sadece bir değişken kullanılabilir.

```
union {
int a;
byte b;
bit [7:0] c;
} my_data;
```

Variable Types

```
module tb;

  union {
    int a;
    byte b;
    bit [7:0] c;
    } my_data;


  initial begin
    my_data.c = 255;
    $display ("%d", my_data.c);
    my_data.b = 100;
    $display ("%d", my_data.c);
  end
endmodule
```

```
# 255
# 100
```

Union Simulasyon Çıktısı

Variable Types

Kullanıcı kendi veri tipini tanımlamak istediğinde typedef yapısını kullanarak tanımlama yapabilmektedir.

```
module tb;
  typedef shortint unsigned u_shorti;
  typedef enum {RED, YELLOW, GREEN} e_light;
  typedef bit [7:0] ubyte;

  initial begin
    u_shorti    data = 32'hface_cafe;
    e_light     light = GREEN;
    ubyte       cnt = 8'hFF;

    $display ("light=%s data=0x%0h cnt=%0d",
light.name(), data, cnt);
  end
endmodule
```

```
# light=GREEN data=0xcafe cnt=255
```

TypeDef