SystemVerilog for Verification

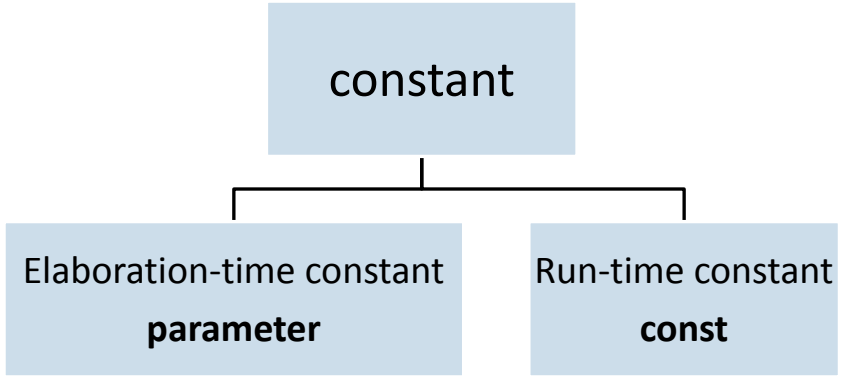
BASIC DATA TYPES – PART IV

Agenda

- ✓ Constant/Parameter
- ✓ Scope & Lifetime
- ✓ Casting

Constant/Parameter

- named data objects that never change.



- **parameter logic** flag = 1 ;

- **module** vector #(size = 1);

  vector v #(3); ← Overwritten size by 3

- **const logic** option = a.b.c ; // acts like a variable that cannot be written

Scope & Lifetime

Scope

File-Scope (C equivalent - global)

- Declaration - *outside* module, interface, program, task, or function
- Accessible anywhere in code

```
int i1;
module a1();
    i1 = 2;          // Ok : In scope
endmodule
module a2();
    i1 = 3;          // Ok : In scope
endmodule
```

Block-Scope (C equivalent - local)

- Declaration - *inside* module, interface, program, checker, task or function
- Accessible within block of code

```
module a1();
    int i1;
    i1 = 2;              // Ok : In scope
endmodule
module a2();
    i1 = 3;          // ERROR : not in scope
endmodule
```

Scope & Lifetime

Lifetime

Static Lifetime

- Exists for the whole simulation
- Keyword – **static** (default mode)

```
for (int i = 0; i < 5; ++i) begin
   static int loop = 0;  //optional static
   loop++;
   $display(loop); // 1 2 3 4 5
end
```

Automatic Lifetime

- lifetime ends when execution leaves their scope, and recreated/reinitialized when the scope is reentered.
- Keyword – **automatic**

```
for (int i = 0; i < 5; ++i) begin
   automatic int loop = 0;
   loop++;
   $display(loop); // 1 1 1 1 1
end
```

Scope & Lifetime

```
int a;
module t1();
  …
endmodule
```

```
module t1();
  int a;
endmodule
```

Scenario	file scope	block scope	static lifetime	automatic lifetime
Variables declared outside a module, program, interface, task, or function	✓		✓	
Variables declared inside a module, interface, program, task, or function		✓	✓	
Variables explicitly declared as automatic in static task, function, or block		✓		✓
Tasks, functions, program - declared as automatic, Variables declared in them		✓		✓
Variables within an automatic task, function, or block can be explicitly declared as static.		✓	✓	

```
task automatic t1();
  int a;
endtask
```

```
task automatic t1();
  int a;
endtask
```

```
task automatic t1();
  static int a;
endtask
```

Casting

Static Casting

- casting operator '
- casting_type ' **(** expression **)**

```
int'(2.0 * 3.0)
shortint'({8'hFA,8'hCE})
signed'(x)
unsigned'(-4);
const'(x)
```

Dynamic Casting

system task - $cast

```
typedef enum { red, green, blue} Colors;
Colors col;
$cast( col, 1 + 2 );

// col=value of enum equivalent to 3 - blue
```

Thank You