

SYSTEM VERILOG

A 2X1 MULTIPLEXER



SYSTEM VERILOG

A 2X1 MULTIPLEXER

This Tutorial covers:

logic — four-state integer type where the four possible states are:

- 1 (high; True);
- 0 (low; False);
- x (unknown) — this means that the state is unknown; Verilog initializes four-state types to X.
- z (high impedance) — this means that the state is disconnected and is essentially “floating.”

output, input — describe the ports associated with a module and provides a way to communicate with a module

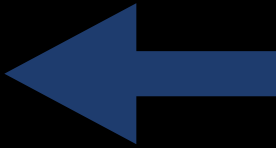
assign — continuous assignment statement in system verilog

module — for simulation of digital circuits in software and synthesis of digital circuits in hardware

SYSTEM VERILOG

A 2X1 MULTIPLEXER

```
module mux2_1(out, i0, i1, sel);  
    output logic out;  
  
    input logic i0, i1, sel;  
  
    assign out = (i1 & sel) | (i0 & ~sel);  
  
endmodule
```



Creates a block of code which can be instantiated (i.e. used) as many times as needed in highboy other files in a design. This module is named mux2_1 and has four ports (out, i0, i1, sel) which allow other modules to communicate with it.

SYSTEM VERILOG

A 2X1 MULTIPLEXER

```
module mux2_1(out, i0, i1, sel);
```

```
    output logic out;
```

```
    input logic i0, i1, sel;
```

```
    assign out = (i1 & sel) | (i0 & ~sel);
```

```
endmodule
```

out is an output of the module mux2_1. out is of the variable type logic and can have four possible values: z, x, 0, or 1

i0, i1, sel are inputs to the module mux2_1. These three inputs are also of the variable type logic and can have four possible values: z, x, 0, or 1

SYSTEM VERILOG

A 2X1 MULTIPLEXER


```
module mux2_1(out, i0, i1, sel);
```

```
    output logic out;
```

```
    input logic i0, i1, sel;
```

```
    assign out = (i1 & sel) | (i0 & ~sel);
```

```
endmodule
```



assigns out
the value of
(i1*sel) + (i0*sel')
where * or & indicates AND;
+ or | indicates OR;
' (~) indicates NOT (INVERT)

SYSTEM VERILOG

A 2X1 MULTIPLEXER

```
module mux2_1_testbench();
```

Creates a block of code that can be used to simulate another module (in this case, mux2_1_testbench() will simulate the mux2_1 created on the preceding slides)

```
    logic i0, i1, sel;  
    logic out;
```

Creates variables i0, i1, sel, and out of variable type logic

```
    mux2_1 dut (.out, .i0, .i1, .sel);
```

Sets up the mux2_1 for testing/simulating and names it dut.

```
    initial begin
```

```
        sel=0; i0=0; i1=0; #10;  
        sel=0; i0=0; i1=1; #10;  
        sel=0; i0=1; i1=0; #10;  
        sel=0; i0=1; i1=1; #10;  
        sel=1; i0=0; i1=0; #10;  
        sel=1; i0=0; i1=1; #10;  
        sel=1; i0=1; i1=0; #10;  
        sel=1; i0=1; i1=1; #10;
```

```
    end
```

```
endmodule
```


SYSTEM VERILOG

A 2X1 MULTIPLEXER

```
module mux2_1_testbench();  
  
    logic i0, i1, sel;  
    logic out;  
  
    mux2_1 dut (.out, .i0, .i1, .sel);  
  
    initial begin  
        sel=0; i0=0; i1=0; #10;  
        sel=0; i0=0; i1=1; #10;  
        sel=0; i0=1; i1=0; #10;  
        sel=0; i0=1; i1=1; #10;  
        sel=1; i0=0; i1=0; #10;  
        sel=1; i0=0; i1=1; #10;  
        sel=1; i0=1; i1=0; #10;  
        sel=1; i0=1; i1=1; #10;  
    end  
  
endmodule
```

This block of code assigns binary values to the three inputs of dut: i0, i1, sel; i0, i1, sel are defined as inputs in the mux2_1 module shown on the preceding slides.

At time = 0: sel i0 i1 = 000

At time = 10 time units: sel i0 i1 = 001

At time = 20 time units: sel i0 i1 = 010

And so on....

SYSTEM VERILOG

A 2X1 MULTIPLEXER

In this introductory tutorial, we have created a 2X1 multiplexer in System Verilog and created a testbench to simulate all possible input combinations in ModelSim.

