User Defined Primitives Part-II

Feb-9-2014

SystemVerilog Migration

Combinational UDPs

In combinational UDPs, the output is determined as a function of the current input. Whenever an input changes value, the UDP is evaluated and one of the state table rows is matched. The output state is set to the value indicated by that row. This is similar to condition statements: each line in table is one condition.

Combinational UDPs have one field per input and one field for the output. Input fields and output fields are separated with colon. Each row of the table is terminated by a semicolon. For example, the following state table entry specifies that when the three inputs are all 0, the output is 0.

```
primitive udp_combo (.....);

table
0 0 0 : 0;
...
endtable

endprimitive
```

You could download file udp_combo.v here

The order of the inputs in the state table description must correspond to the order of the inputs in the port list in the UDP definition header. It is not related to the order of the input declarations.

Each row in the table defines the output for a particular combination of input states. If all inputs are specified as x, then the output must be specified as x. All combinations that are not explicitly specified result in a default output state of x.

Example

In the below example entry, the ? represents a don't-care condition. This symbol indicates iterative substitution of 1, 0, and x. The table entry specifies that when the inputs are 0 and 1, the output is 1 no matter what the value of the current state is.

You do not have to explicitly specify every possible input combination. All combinations that are not explicitly specified result in a default output state of x.

It is illegal to have the same combination of inputs, specified for different outputs.

```
// This code shows how UDP body looks like
primitive udp_body (
a, // Port a
b, // Port b
c  // Port c
);
output a;
input b,c;

// UDP function code here
// A = B | C;
table
 // B  C    : A
     ?  1    : 1;
     1  ?    : 1;
     0  0    : 0;
endtable

endprimitive
```

You could download file udp_body.v here

TestBench to check above UDP

```
`include "udp_body.v"
module udp_body_tb();

reg b,c;
wire a;

udp_body udp (a,b,c);

initial begin
  $monitor(" B = %b C = %b  A = %b",b,c,a);
  b = 0;
  c = 0;
  #1  b = 1;
  #1  b = 0;
  #1  c = 1;
  #1  b = 1'bx;
  #1  c = 0;
  #1  b = 1;
  #1  c = 1'bx;
  #1  b = 0;
  #1  $finish;
end

endmodule
```

You could download file udp_body_tb.v here

Simulator Output

```
 B = 0 C = 0  A = 0
 B = 1 C = 0  A = 1
 B = 0 C = 0  A = 0
 B = 0 C = 1  A = 1
 B = x C = 1  A = 1
 B = x C = 0  A = x
 B = 1 C = 0  A = 1
 B = 1 C = x  A = 1
 B = 0 C = x  A = x
```

BACK HOME NEXT