User Defined Primitives Part-III

Feb-9-2014

SystemVerilog Migration

Level Sensitive Sequential UDP

Level-sensitive sequential behavior is represented in the same way as combinational behavior, except that the output is declared to be of type reg, and there is an additional field in each table entry. This new field represents the current state of the UDP.

- The output is declared as reg to indicate that there is an internal state. The output value of the UDP is always the same as the internal state.
- A field for the current state has been added. This field is separated by colons from the inputs and the output.

Sequential UDPs have an additional field inserted between the input fields and the output field, compared to combinational UDP. This additional field represents the current state of the UDP and is considered equivalent to the current output value. It is delimited by colons.

```
1 primitive udp_seq (.....);
2
3 table
4 0 0 0 : 0 : 0;
5 ...
6 endtable
7
8 endprimitive
```

You could download file udp_seq.v here

Example

```
1 primitive udp_latch(q, clk, d) ;
2 output q;
3 input clk, d;
4
5 reg q;
6
7 table
8   //clk d   q    q+
```

```
 9   0     1  : ? :   1   ;
10   0     0  : ? :   0   ;
11   1     ?  : ? :   -   ;
12 endtable
13
14 endprimitive
```

You could download file udp_latch.v here

Edge-Sensitive UDPs

In level-sensitive behavior, the values of the inputs and the current state are sufficient to determine the output value. Edge-sensitive behavior differs in that changes in the output are triggered by specific transitions of the inputs.

As in the combinational and the level-sensitive entries, a ? implies iteration of the entry over the values 0, 1, and x. A dash (-) in the output column indicates no value change.

All unspecified transitions default to the output value x. Thus, in the previous example, transition of clock from 0 to x with data equal to 0 and current state equal to 1 result in the output q going to x.

All transitions that should not affect the output must be explicitly specified. Otherwise, they will cause the value of the output to change to x. If the UDP is sensitive to edges of any input, the desired output state must be specified for all edges of all inputs.

Example

```
 1 primitive udp_sequential(q, clk, d);
 2 output q;
 3 input clk, d;
 4
 5 reg q;
 6
 7 table
 8 // obtain output on rising edge of clk
 9 // clk      d     q     q+
10    (01)        0  :  ?  :  0  ;
11    (01)        1  :  ?  :  1  ;
12    (0?)        1  :  1  :  1  ;
13    (0?)        0  :  0  :  0  ;
14 // ignore negative edge of clk
15    (?0)        ?  :  ?  :  -  ;
16 // ignore d changes on steady clk
17    ?     (??)     :  ?  :  -  ;
18  endtable
19
20 endprimitive
```

You could download file udp_sequential.v here

Example UDP with initial

```
primitive udp_sequential_initial(q, clk, d);
output q;
input clk, d;

reg q;

initial begin
  q = 0;
end

table
// obtain output on rising edge of clk
// clk      d     q     q+
   (01)        0  :  ?  :  0  ;
   (01)        1  :  ?  :  1  ;
   (0?)        1  :  1  :  1  ;
   (0?)        0  :  0  :  0  ;
// ignore negative edge of clk
   (?0)        ?  :  ?  :  -  ;
// ignore d changes on steady clk
   ?     (??)     :  ?  :  -  ;
 endtable

endprimitive
```

You could download file udp_sequential_initial.v here