

2001 IC/CAD Contest

Problem 7: User-Defined Primitive (UDP) modeling

Source: SynTest Technologies, Inc.

Dec. 20, 2000

I. Introduction

The UDP is a special primitive supported by the Verilog language. UDP stands for "User-Defined Primitive" whose functionality can be defined by users by means of writing a table like description. Both combinational and sequential behavior can be modeled with UDPs. UDPs are mostly used in library to model a set of pre-defined gate primitives.

For detail description of the UDP, please refer to the Cadence Verilog-XL reference manual.

UDP provides an efficient way to describe the gate primitives behavior, and can improve simulation performance. However, in some CAD applications, Verilog gate primitives (and, or, not, etc ...) and procedural constructs are better inputs.

In most cases, a UDP can be rewritten with Verilog gate primitives and/or procedural constructs. For example, the combinational UDP "mux_udp" can also be modeled with the primitives in module "mux" below :

```
primitive mux_udp(Out, A, B, Sel)
output Out;
input  A, B, Sel;
table
// A  B  Sel  Out
  0  ?  0  :  0;
  1  ?  0  :  1;
  x  ?  0  :  x;
  ?  0  1  :  0;
  ?  1  1  :  1;
  ?  x  1  :  x;
  0  0  x  :  0;
  1  1  x  :  1;
endtable
endprimitive
```

```

module mux(Out, A, B, Sel);
output Out;
input  A, B, Sel;

    not  (w0, Sel);
    and  (w1, A, w0);
    and  (w2, B, Sel);
    and  (w3, A, B);
    or   (Out, w1, w2, w3);

endmodule

```

The following sequential UDP "dff_udp" can also be modeled with the module "dff_mod" :

```

primitive dff_udp (q, clk, d)
input clk, d;
output q;

reg q;

table
// clk d  q  q+
(01) 0  :  ?  :  0;
(01) 1  :  ?  :  1;
(10) ?  :  ?  :  -;
?    *  :  ?  :  -;
endtable
endprimitive

module udp_mod (q, clk, d)
input clk, d;
output q;

reg q;
always @(posedge clk)
    q <= d;

endmodule

```

II. Problem Description

Given a UDP primitive, develop an algorithm to translate the UDP primitive into a functionally equivalent module consisting of only the Verilog gate primitives and/or procedural statements.

III. Input and output

Input:

The input is a file containing Verilog UDP definition. Examples of UDP definition are given below:

1. Combinational UDP

```
primitive Comb (Y, D0, D1, D2, D3, S0, S1);  
  input D0, D1, D2, D3, S0, S1;  
  output Y;
```

```
  table
```

```
//   D0  D1  D2  D3  S0 S1 : Y  
  
    0   ?   ?   ?   0  0  : 0 ;  
    1   ?   ?   ?   0  0  : 1 ;  
    ?   0   ?   ?   1  0  : 0 ;  
    ?   1   ?   ?   1  0  : 1 ;  
    ?   ?   0   ?   0  1  : 0 ;  
    ?   ?   1   ?   0  1  : 1 ;  
    ?   ?   ?   0   1  1  : 0 ;  
    ?   ?   ?   1   1  1  : 1 ;  
    0   0   0   0   ?   ?  : 0 ;  
    1   1   1   1   ?   ?  : 1 ;  
    0   0   ?   ?   ?   0  : 0 ;  
    1   1   ?   ?   ?   0  : 1 ;  
    ?   ?   0   0   ?   1  : 0 ;  
    ?   ?   1   1   ?   1  : 1 ;  
    0   ?   0   ?   0   ?  : 0 ;  
    1   ?   1   ?   0   ?  : 1 ;  
    ?   0   ?   0   1   ?  : 0 ;  
    ?   1   ?   1   1   ?  : 1 ;
```

```
  endtable  
endprimitive
```

2. Sequential UDP

```
primitive Sequential (QN, J, K, CP, CD, SD, notifier);
  output QN;
  reg QN;
  input J, K, CP, CD, SD, notifier;
```

table	// J	K	CP	CD	SD	notifier	: Qtn	: Qtn+1	
0	0	(01)	1	1		?	:	?	-;
0	0	?	1	?		?	:	0	-;
0	1	p	?	1		?	:	?	1;
1	0	p	1	?		?	:	?	0;
?	0	?	1	?		?	:	0	-;
?	1	(01)	?	1		?	:	0	1;
1	?	(01)	1	?		?	:	1	0;
0	0	(x1)	1	1		?	:	?	-;
0	0	(0x)	1	1		?	:	?	-;
*	?	1	?	1		?	:	1	-;
*	?	0	?	1		?	:	1	-;
?	*	1	?	1		?	:	1	-;
?	*	0	?	1		?	:	1	-;
*	?	1	1	?		?	:	0	-;
*	?	0	1	?		?	:	0	-;
?	*	1	1	?		?	:	0	-;
?	*	0	1	?		?	:	0	-;
*	?	1	1	1		?	:	?	-;
*	?	0	1	1		?	:	?	-;
?	*	1	1	1		?	:	?	-;
?	*	1	1	1		?	:	?	-;
?	*	1	1	1		?	:	?	-;
?	*	0	1	1		?	:	?	-;
?	?	?	?	0		?	:	?	0;
?	?	?	0	1		?	:	?	1;
?	?	n	1	1		?	:	?	-;
?	?	n	?	1		?	:	1	-;
?	?	n	1	?		?	:	0	-;
?	?	0	*	1		?	:	1	-;
?	?	1	*	1		?	:	1	-;
?	?	0	1	*		?	:	0	-;
?	?	1	1	*		?	:	0	-;
?	?	?	?	?		*	:	?	x;

```
endtable  
endprimitive
```

Output

1. Verilog netlist. If the input pin the UDP has no effects on the functionality, it can be removed from the module input list.
2. Verilog simulation test bench to verify the results.

IV. Language/Platform

Language : C

Platform : Linux or Sun/Solaris

V. Evaluation

1. Novelty of the algorithm.
2. Functional equivalence.
3. CPU time used.

VI. References

1. Cadence Verilog-XL Reference Manual.
2. The Verilog Hardware Description Language.
By Thomas & Moorby's
KAP (Kluwer Academic Publishers)