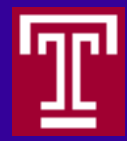
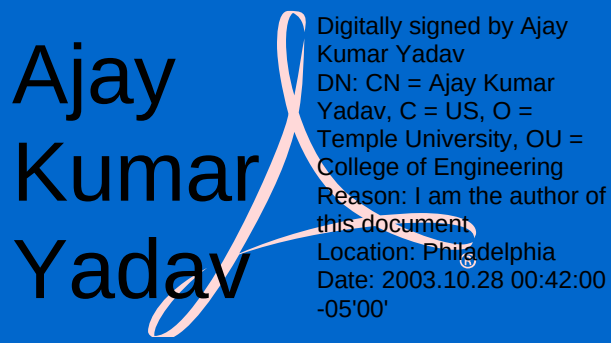
EE335: Advanced Microprocessor



Chapter 4

Introduction to Logic Design with Verilog



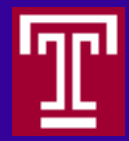
Ajay Kumar Yadav

(Instructor)

Electrical & Computer Engineering

Temple University

Verilog Primitives:



Basic Functional Objects that can be used to design a circuit. (Model of combinational Logic Gates)

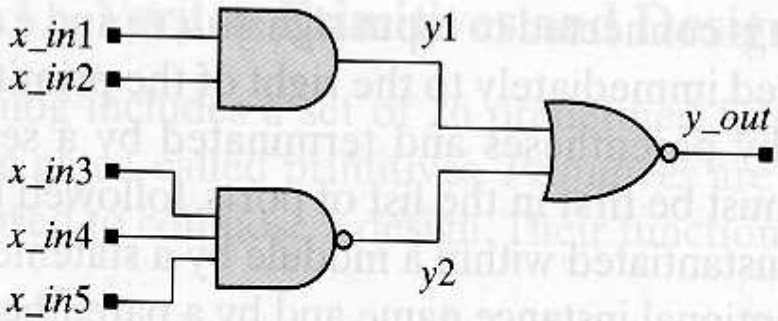
n-Input	*n-output, 3-state*
and	buf
nand	not
or	bufif0
nor	bufif1
xor	notif0
xnor	notif1

****Verilog includes a set of 26 predefined models****

Output port must be first, instance name is optional

Example:

5-Input AOI Circuit




```
wire    y1, y2;
nor     (y_out, y1, y2);
and     (y1, x_in1, x_in2);
nand    (y2, x_in3, x_in4, x_in5);
```

FIGURE 4-3 A list of declarations of wire-connected primitives having the functionality of a five-input AOI gate.

Module Format:

Module name(module ports)

……// Declaration of ports, registers, wires

……// Functional detail

Endmodule

- Structural
- Behavioral
- Combination of both

Structural Model:

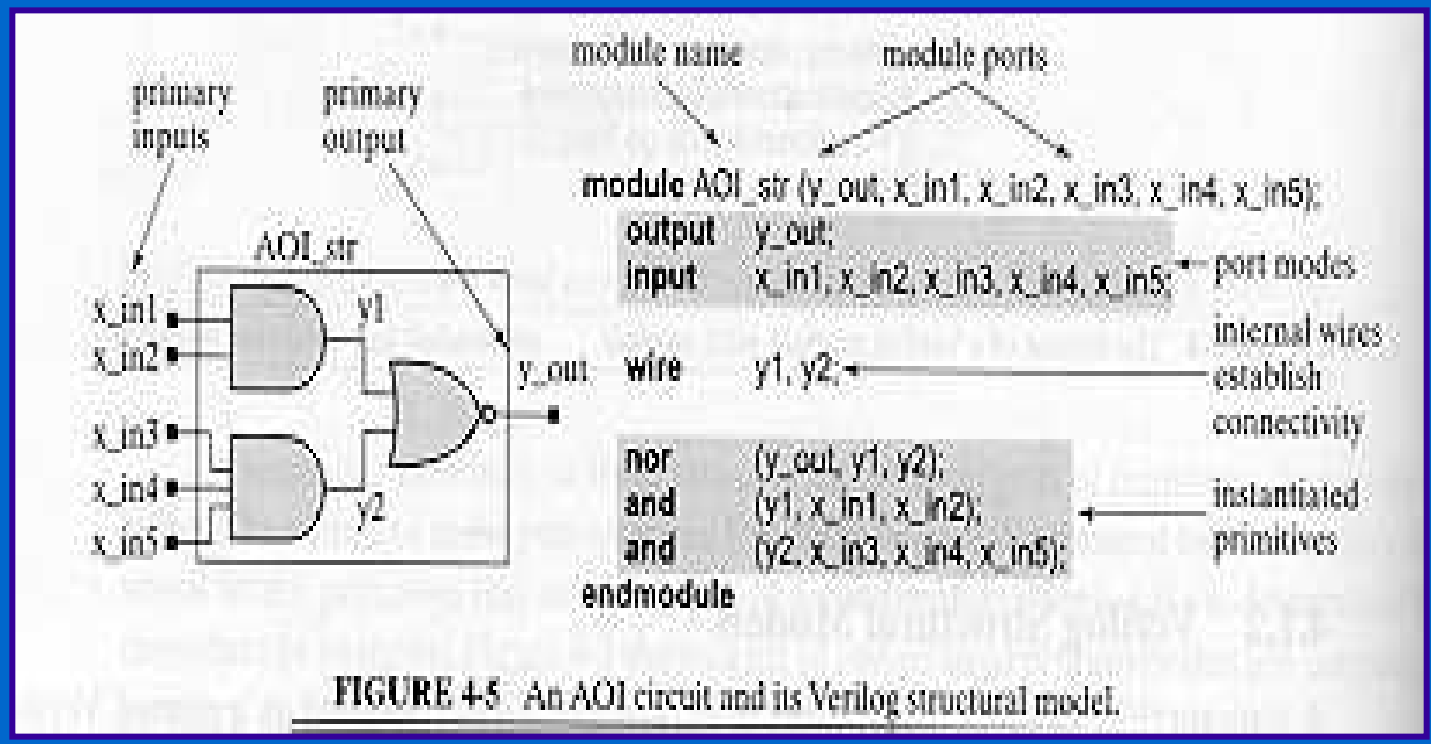


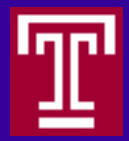
FIGURE 4-5 An AOI circuit and its Verilog structural model.

****Type of Ports: input, output, inout****

Language Rule:

- Case Sensitive.
- Identifier (no space, apart from alphabets, digits $ and _ is permitted.)
- First character of variable can neither be digit or $
- Line termination “ ;” is must , except *endmodule*.
- // text is considered as comment.
- /*, */ pair is used for multiple line comment.

Top-Down Design & Nested Module:



- Top-Down: Break and rule (generally used for complex and higher level architecture).
- Nested Module: Instantiation of a module within the declaration of different module.

Design Hierarchy(16-bit Ripple-carry adder):

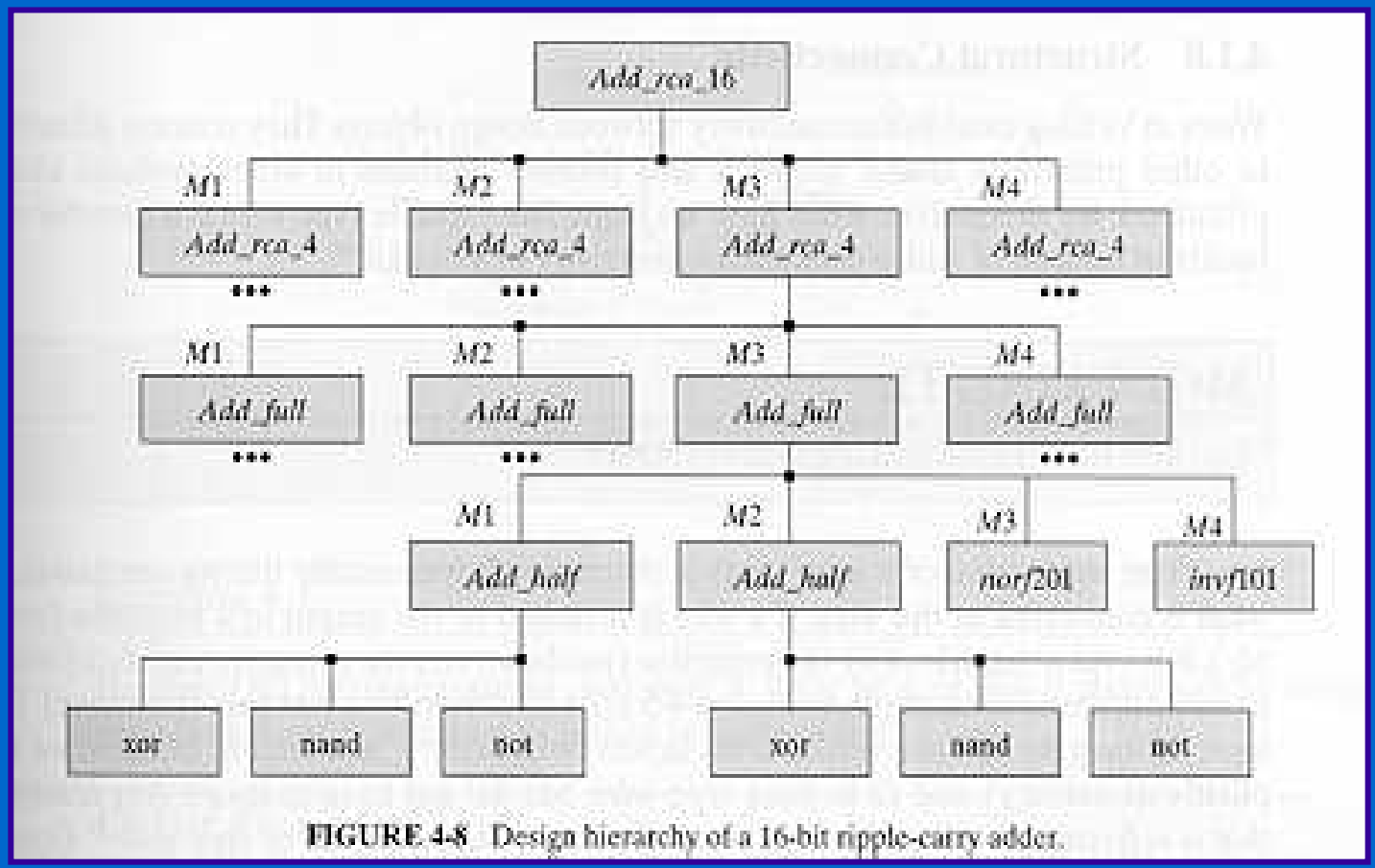


FIGURE 4-8 Design hierarchy of a 16-bit ripple-carry adder.

Hierarchical Decomposition:

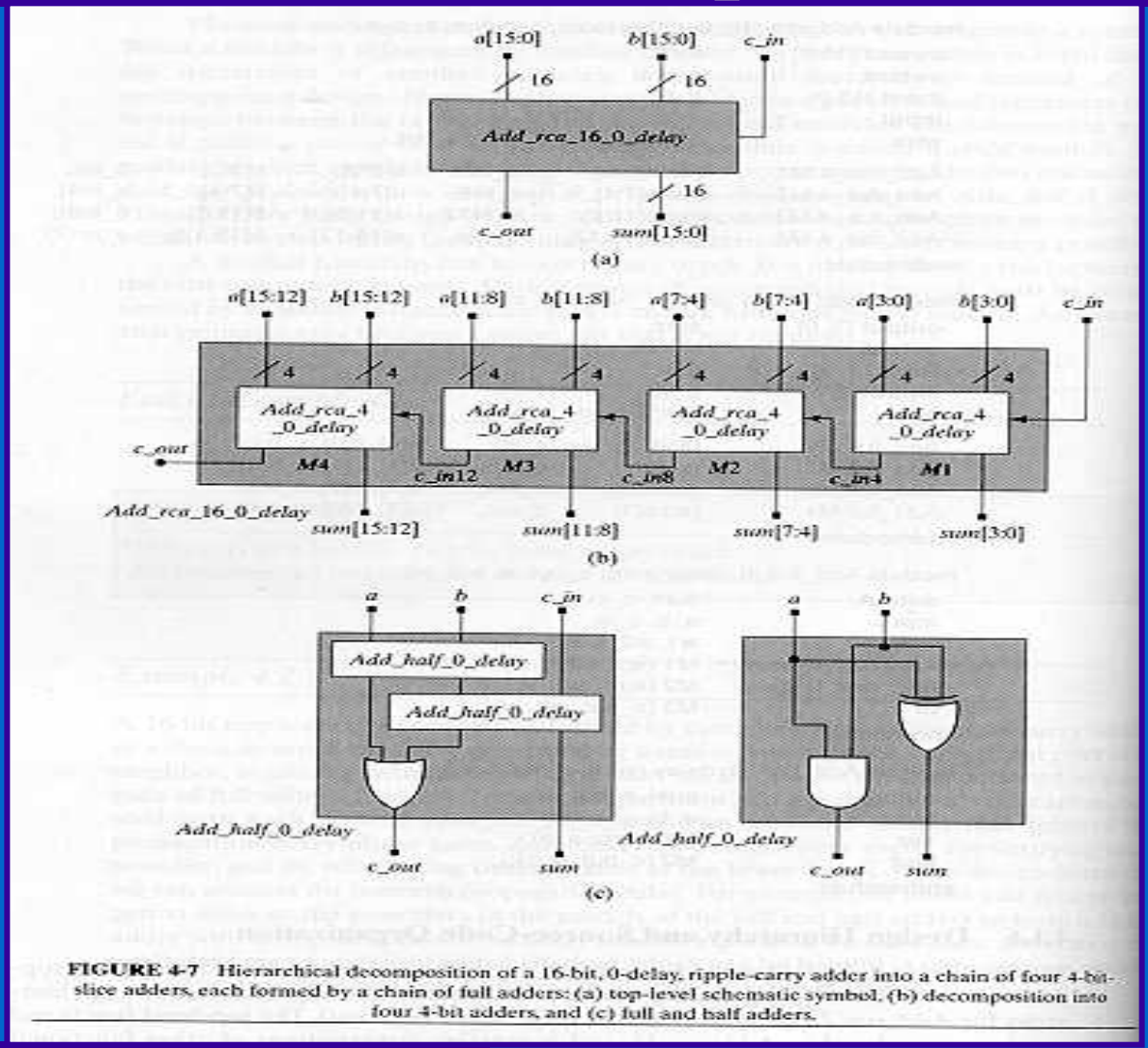


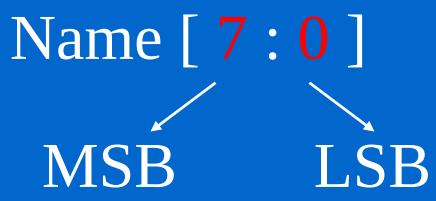
FIGURE 4-7 Hierarchical decomposition of a 16-bit, 0-delay, ripple-carry adder into a chain of four 4-bit-slice adders, each formed by a chain of full adders: (a) top-level schematic symbol, (b) decomposition into four 4-bit adders, and (c) full and half adders.

Verilog Code:

```
module Add_rca_16_0_delay (sum, c_out, a, b, c_in);
  output [15:0]     sum;
  output            c_out;
  input [15:0]      a, b;
  input             c_in;
  wire              c_in4, c_in8, c_in12, c_out;

  Add_rca_4 M1      (sum[3:0],    c_in4,    a[3:0],    b[3:0],    c_in);
  Add_rca_4 M2      (sum[7:4],    c_in8,    a[7:4],    b[7:4],    c_in4);
  Add_rca_4 M3      (sum[11:8],   c_in12,   a[11:8],   b[11:8],   c_in8);
  Add_rca_4 M4      (sum[15:12],  c_out,    a[15:12],  b[15:12],  c_in12);
endmodule

module Add_rca_4 (sum, c_out, a, b, c_in);
  output [3: 0]     sum;
  output            c_out;
  input [3: 0]      a, b;
  input             c_in;
  wire              c_in2, c_in3, c_in4;

  Add_full M1       (sum[0],  c_in2,  a[0], b[0], c_in);
  Add_full M2       (sum[1],  c_in3,  a[1], b[1], c_in2);
  Add_full M3       (sum[2],  c_in4,  a[2], b[2], c_in3);
  Add_full M4       (sum[3],  c_out,  a[3], b[3], c_in4);
endmodule

module Add_full_0_delay(sum, c_out, a, b, c_in);
  output            sum, c_out;
  input             a, b, c_in;
  wire              w1, w2, w3;
  Add_half_0_delay  M1 (w1, w2, a, b);
  Add_half_0_delay  M2 (sum, w3, w2, c_in);
  or                M3 (c_out, w2, w3);
endmodule

module Add_half_0_delay (sum, c_out, a, b);
  output            sum, c_out;
  input             a, b;
  xor               M1 (sum, a, b);
  and               M2 (c_out, a, b);
endmodule
```

Vectors in Verilog:

Denoted by square brackets, enclosing the range of the bits.



Structural Connectivity:

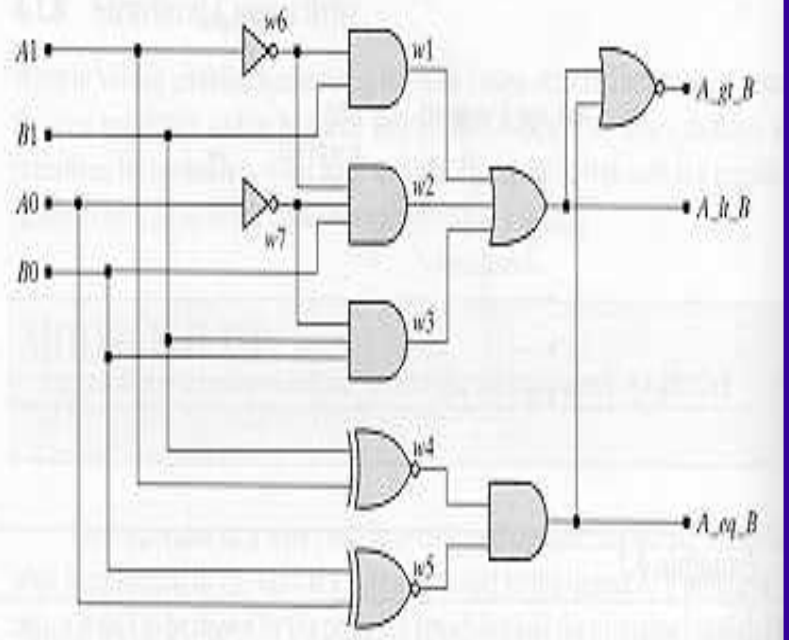


FIGURE 4-10 Schematic of a 2-bit binary comparator.

```
module compare_2_str (A_gt_B, A_lt_B, A_eq_B, A0, A1, B0, B1);
  output   A_gt_B, A_lt_B, A_eq_B;
  input    A0, A1, B0, B1;

  nor      (A_gt_B, A_lt_B, A_eq_B);
  or       (A_lt_B, w1, w2, w3);
  and      (A_eq_B, w4, w5);
  and      (w1, w6, B1);
  and      (w2, w6, w7, B0);
  and      (w3, w7, B0, B1);
  not      (w6, A1);
  not      (w7, A0);
  xnor     (w4, A1, B1);
  xnor     (w5, A0, B0);
endmodule
```

Undeclared identifiers are treated as wire by default

Design Verification:

Verification is necessary to assure the reliable functionality of the design or designed system

Two methods of verification:

1. Logic Simulation: Stimulus patterns are applied to monitor the behavioral simulation of the design
2. Formal Verification: Uses elaborate mathematical proofs to verify a circuit's functionality.

Value Logic:

Verilog supports four valued logic system:

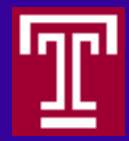
0, 1, x, z.

0, 1 corresponds to False or true of a signal, respectively.

x indicates the state of ambiguity (two opposite values are provided to a same wire).

z indicates high impedance state (connection failure).

Test-bench Template:



```
module some_name ();
reg  //declaration of primary input of the UUT
wire // declaration of primary output of the UUT
UUT_name instance_name (UUT I/O ports)
initial $monitor (); //Specification of signals to be monitored
initial #time $stop; // termination of simulation
initial //pattern generation and/or error detection
begin
    //behavioral statements generating wf to the input ports
   //uses the full repertoire of behavioral constructs for loops and conditions.
end
endmodule
```

Inertial Delay:

Time interval in which a input pulse must be constant in order for the gate to make a transition.

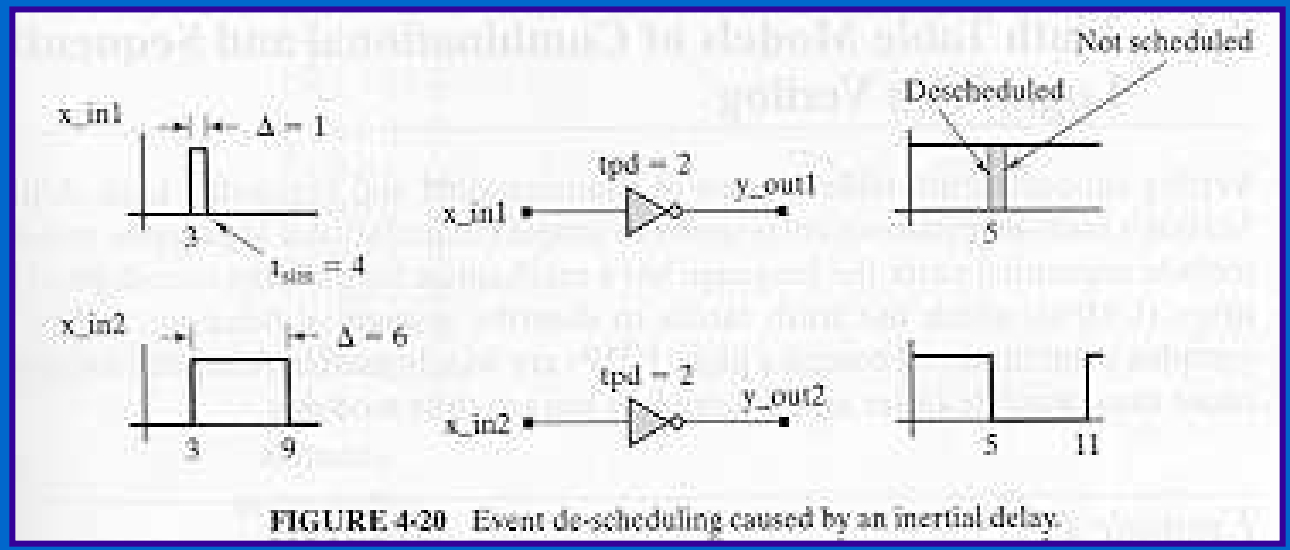


FIGURE 4-20 Event de-scheduling caused by an inertial delay.

Transport Delay:

Time of flight of a signal traveling a wire of a circuit

In such models, transitions are not suppressed and all the transitions occur on the receiver end after a finite time delay.

*** Delays in ASIC Design -> 0.033ns / cm. ***

Truth Table Model:

TTM supports both combinational and sequential logic. Generally used to design user defined primitives (UDP).

****Built in primitives correspond to combinational logic.****

Rules for UDP:

1. Output and inputs must be scalar.
2. Single output is permitted.
3. All the possible combinations need to be defined.

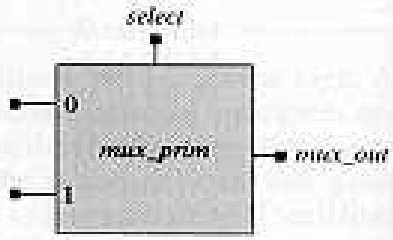
Application: Mostly in ASIC cell libraries.

Advantage: Simulate faster and requires less storage.

AI (And Invert) Circuit:


```
primitive AI (o, a, b) ;
output o ;
input a, b;
table
        0  0 : 1 ;
        0  1 : 1 ;
        1  0 : 1 ;
        1  1 : 0 ;
endtable
endprimitive
```

UDP for 2-input MUX:




```
primitive mux_prim (mux_out, select, a, b);
  output mux_out;
  input  select, a, b;
  table

// select     a     b     :  mux_out

     0        0     0     :     0 ;     // Order of table columns 5 port order of inputs
     0        0     1     :     0 ;     // One output, multiple inputs, no inout
     0        0     x     :     0 ;     // Only 0, 1, x on input and output
     0        1     0     :     1 ;     // A z input in simulation is treated as x
     0        1     1     :     1 ;     // by the simulator
     0        1     x     :     1 ;     // Last column is the output

// select     a     b     :  mux_out
     1        0     0     :     0 ;
     1        1     0     :     0 ;
     1        x     0     :     0 ;

     1        0     1     :     1 ;
     1        1     1     :     1 ;
     1        x     1     :     1 ;

     x        0     0     :     0 ;     // Reduces pessimism
     x        1     1     :     1 ;

  endtable          // Note: Combinations not explicitly specified will drive 'x'
endprimitive        // under simulation.
```

FIGURE 4-21 UDP for a two-input multiplexer.

Implementation of Shorthand notation:

```
table
// Shorthand notation:
// ? represents iteration of the table entry over the values 0,1,x.
// i.e., don't care on the input
//    select    a    b    : mux_out
//    0         0    ?    :      0; // ? = 0,1,x shorthand notation.
//    0         1    ?    :      1;
//    1         ?    0    :      0;
//    1         ?    1    :      1;
//    ?         0    0    :      0;
//    ?         1    1    :      1;
endtable
```

*** ? : represents value 0,1,x ***