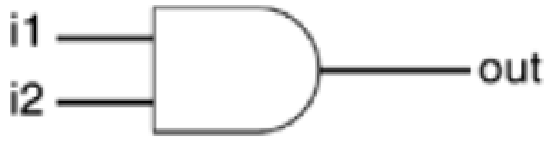
Gate-Level Modeling or Structural Modeling

Gate Level

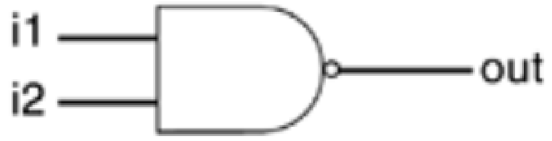
- Four levels of abstraction are used to describe the hardware.
- Gate level is low level of abstraction
- The circuit is described in terms of gates (e.g., and, nand, …)
- Verilog description in gate level corresponds to logic circuit diagram
- Verilog supports basic logic gates as predefined primitives
- These primitives are instantiated like modules except that they are predefined in Verilog and do not need a module definition
- There are two classes of basic gates: and/or gates and buf/not gates

Gate-Level Modeling or Structural Modeling

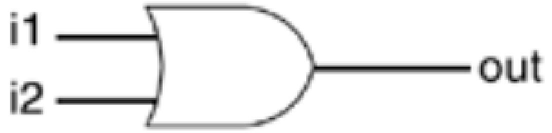
Gate Types – and/or gates



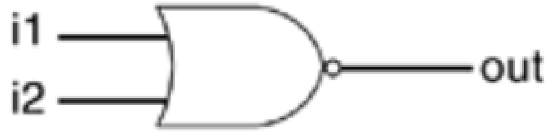
and



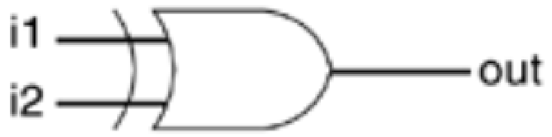
nand



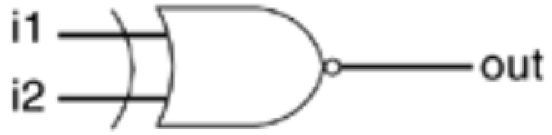
or



nor



xor



xnor

Gate-Level Modeling or Structural Modeling

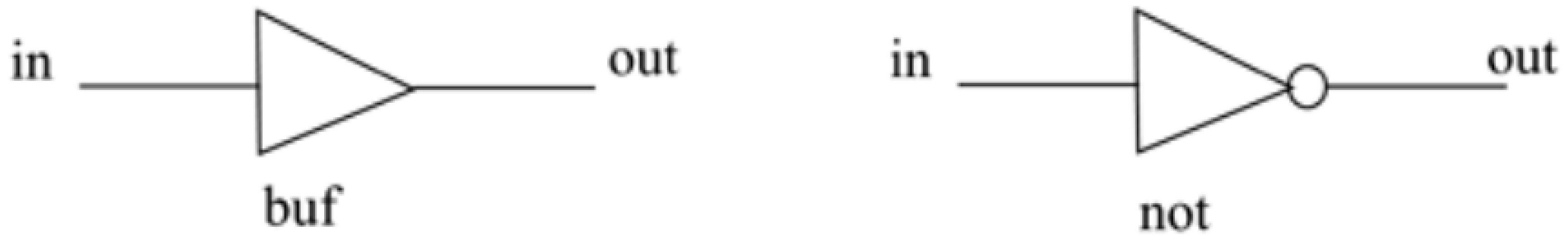
Gate Instantiation of And/Or gates

- And/or gates have one scalar output and multiple scalar inputs.
- The first terminal in the list of gate terminals is an output and the other terminals are inputs.
- The output of a gate is evaluated as soon as one of the inputs changes.
- instance name does not need to be specified for primitives

```
wire OUT, IN1, IN2;
// basic gate instantiations
and a1(OUT, IN1, IN2);
nand na1(OUT, IN1, IN2);
or or1(OUT, IN1, IN2);
nor nor1(OUT, IN1, IN2);
xor x1(OUT, IN1, IN2);
xnor nx1(OUT, IN1, IN2);
// More than two inputs; 3 input nand gate
nand na1_3inp(OUT, IN1, IN2, IN3);
// gate instantiation without instance name
and (OUT, IN1, IN2); // legal gate instantiation
```

Gate-Level Modeling or Structural Modeling

Gate Types – Buf/Not gates



Gate-Level Modeling or Structural Modeling

Gate Instantiation of Buf/Not gates

- And/or gates have one scalar input and one or more scalar outputs.
- The last terminal in the port list is connected to the input. Other terminals are connected to the outputs.
- The output of a gate is evaluated as soon as one of the inputs changes.
- instance name does not need to be specified for primitives

```
// basic gate instantiations.
buf b1(OUT1, IN);
not n1(OUT1, IN);

// More than two outputs
buf b1_2out(OUT1, OUT2, IN);

// gate instantiation without instance name
not (OUT1, IN); // legal gate instantiation
```

buf	in	out
	0	0
	1	1
	x	x
	z	x

not	in	out
	0	1
	1	0
	x	x
	z	x

Gate-Level Modeling or Structural Modeling

Gate Instantiation of Bufif/notif gates

- Gates with an additional control signal on buf and not gates
- These gates propagate only if their control signal is stated
- They propagate z if their control signal is not stated

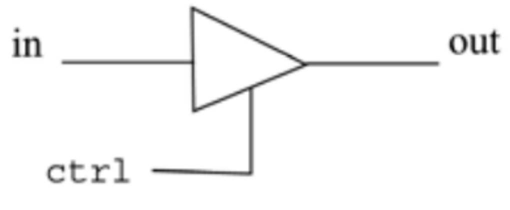
```
//Instantiation of bufif gates
bufif1 b1 (out, in, ctrl);
bufif0 b0 (out, in, ctrl);
//Instantiation of notif gates
notif1 n1 (out, in, ctrl);
notif0 n0 (out, in, ctrl);
```

		ctrl			
	bufif1	0	1	x	z
in	0	z	0	L	L
	1	z	1	H	H
	x	z	x	x	x
	z	z	x	x	x

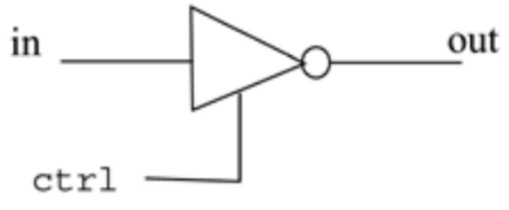
		ctrl			
	bufif0	0	1	x	z
in	0	0	z	L	L
	1	1	z	H	H
	x	x	z	x	x
	z	x	z	x	x

		ctrl			
	notif1	0	1	x	z
in	0	z	1	H	H
	1	z	0	L	L
	x	z	x	x	x
	z	z	x	x	x

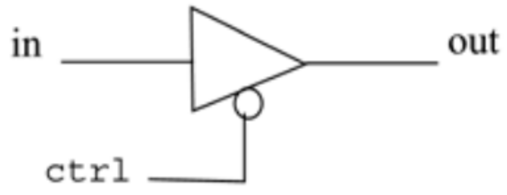
		ctrl			
	notif0	0	1	x	z
in	0	1	z	H	H
	1	0	z	L	L
	x	x	z	x	x
	z	x	z	x	x



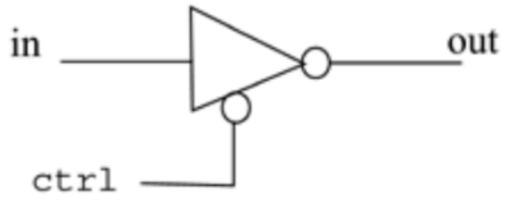
bufif1



notif1



bufif0



notif0

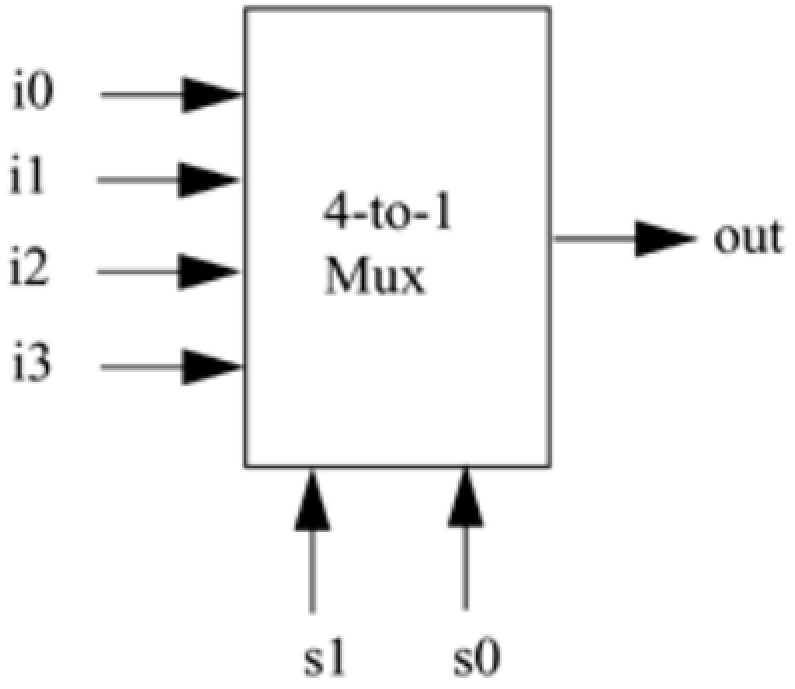
Gate-Level Modeling or Structural Modeling

Array of primitive Gate Instantiation

```
wire [7:0] OUT, IN1, IN2;

// basic gate instantiations
nand n_gate[7:0](OUT, IN1, IN2);

// This is equivalent to the following 8 instantiations
nand n_gate0(OUT[0], IN1[0], IN2[0]);
nand n_gate1(OUT[1], IN1[1], IN2[1]);
nand n_gate2(OUT[2], IN1[2], IN2[2]);
nand n_gate3(OUT[3], IN1[3], IN2[3]);
nand n_gate4(OUT[4], IN1[4], IN2[4]);
nand n_gate5(OUT[5], IN1[5], IN2[5]);
nand n_gate6(OUT[6], IN1[6], IN2[6]);
nand n_gate7(OUT[7], IN1[7], IN2[7]);
```

Gate-Level Modeling or Structural Modeling

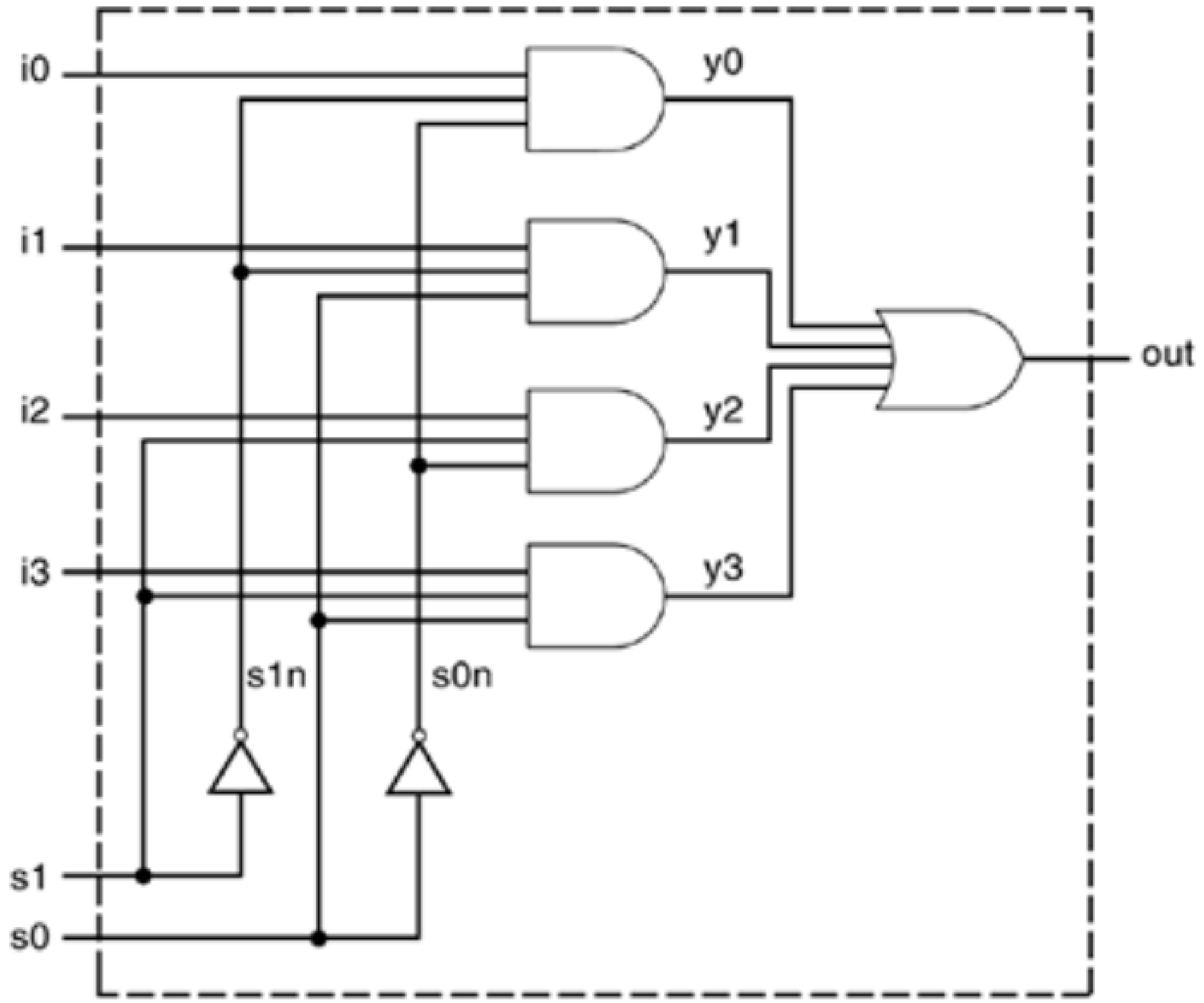
4-to-1 Multiplexer



s1	s0	out
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Gate-Level Modeling or Structural Modeling

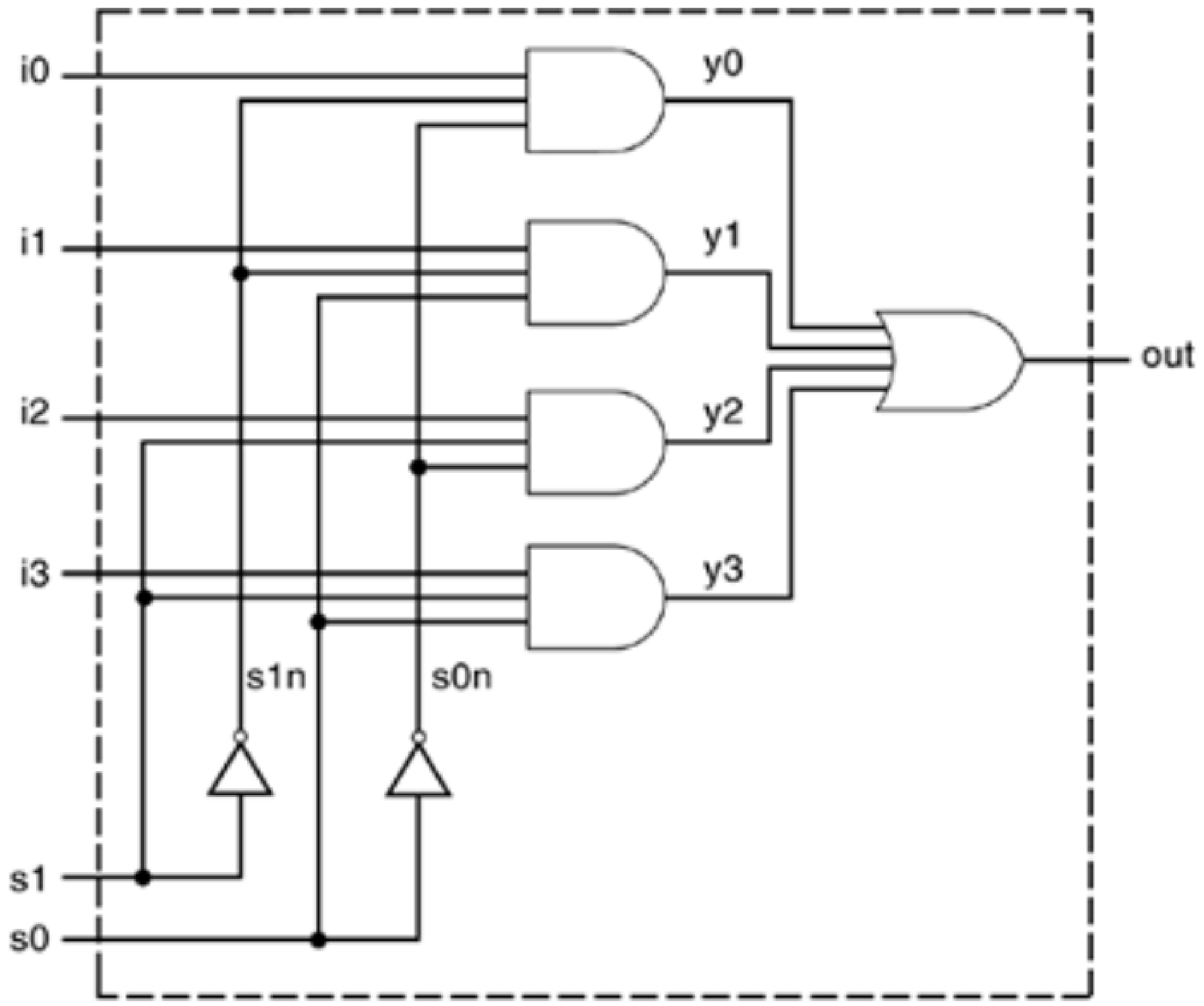
4-to-1 Multiplexer



Gate-Level Modeling or Structural Modeling

Verilog Description of Multiplexer

```
module mux4_to_1 (out, i0, i1, i2, i3, s1, s0);
// Port declarations from the I/O diagram
output out;
input i0, i1, i2, i3;
input s1, s0;
// Internal wire declarations
wire s1n, s0n;
wire y0, y1, y2, y3;
// Gate instantiations
// Create s1n and s0n signals.
not (s1n, s1);
not (s0n, s0);
// 3-input and gates instantiated
and (y0, i0, s1n, s0n);
and (y1, i1, s1n, s0);
and (y2, i2, s1, s0n);
and (y3, i3, s1, s0);
// 4-input or gate instantiated
or (out, y0, y1, y2, y3);
endmodule
```



Gate-Level Modeling or Structural Modeling

Verilog Description of Multiplexer Testbench

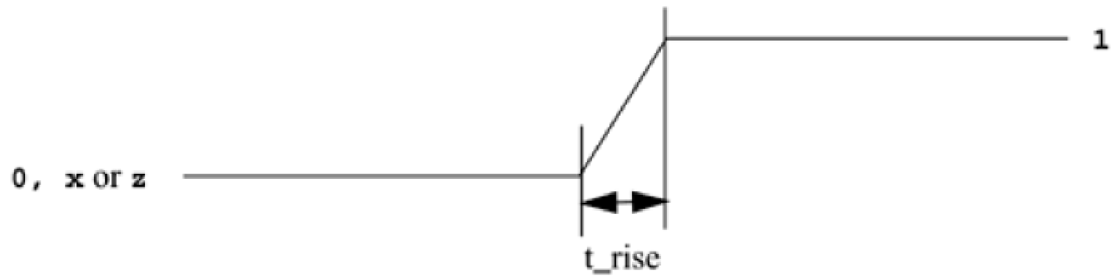
```
module stimulus;
reg IN0, IN1, IN2, IN3;
reg S1, S0;
wire OUTPUT;
mux4_to_1 mymux(OUTPUT, IN0, IN1, IN2, IN3, S1, S0);
initial
begin
IN0 = 1; IN1 = 0; IN2 = 1; IN3 = 0;
#1 $display("IN0= %b, IN1= %b, IN2= %b, IN3= %b\n",IN0,IN1,IN2,IN3);
S1 = 0; S0 = 0;
#1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUTPUT);
S1 = 0; S0 = 1;
#1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUTPUT);
S1 = 1; S0 = 0;
#1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUTPUT);
S1 = 1; S0 = 1;
#1 $display("S1 = %b, S0 = %b, OUTPUT = %b \n", S1, S0, OUTPUT);
end
endmodule
```

Output

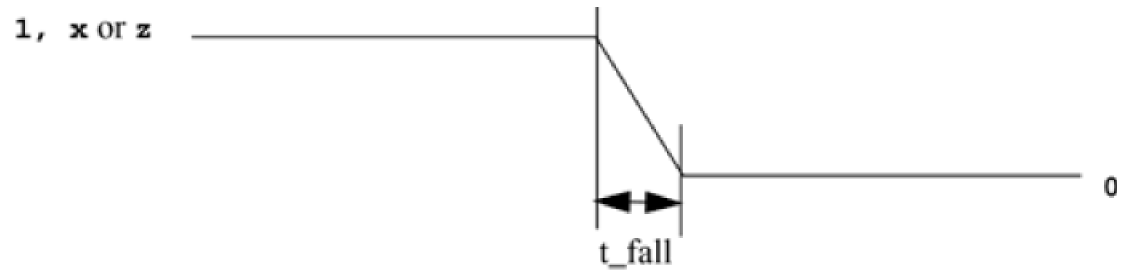
```
IN0= 1, IN1= 0, IN2= 1, IN3= 0

S1 = 0, S0 = 0, OUTPUT = 1
S1 = 0, S0 = 1, OUTPUT = 0
S1 = 1, S0 = 0, OUTPUT = 1
S1 = 1, S0 = 1, OUTPUT = 0
```

Gate-Level Modeling - Gate delays

- In real circuits, logic gates have delays associated with them
- Gate delays allow the Verilog user to specify delays through the logic circuits
- If no delays are specified, the default value is zero.
- Three types of delays from the inputs to the output of a primitive gate
    - Rise delay



    - Fall delay



    - Turn-off delay
        - The turn-off delay is associated with a gate output transition to the high impedance value (z) from another value

Gate-Level Modeling or Structural Modeling

Types of Delay Specification Syntax

```
// Delay of delay_time for all transitions
and #(delay_time) a1(out, i1, i2);
// Rise and Fall Delay Specification.
and #(rise_val, fall_val) a2(out, i1, i2);
// Rise, Fall, and Turn-off Delay Specification
bufif0 #(rise_val, fall_val, turnoff_val) b1 (out, in, control);
```

Examples of delay specification

```
and #(5) a1(out, i1, i2); //Delay of 5 for all transitions
and #(4,6) a2(out, i1, i2); // Rise = 4, Fall = 6
bufif0 #(3,4,5) b1 (out, in, control); // Rise = 3, Fall = 4, Turn-off = 5
```

Gate-Level Modeling or Structural Modeling

Verilog Description of 4-bit Ripple Carry full adder Testbench

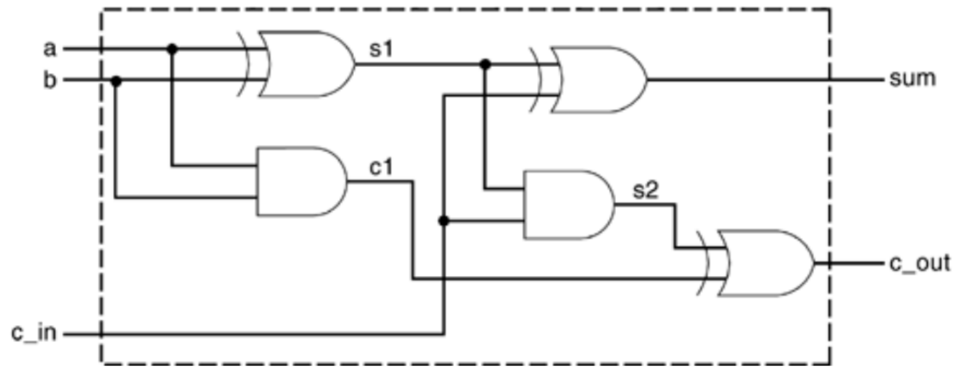
```
module stimulus; // Define the stimulus (top level module)
reg [3:0] A, B; // Set up variables
reg C_IN;
wire [3:0] SUM;
wire C_OUT;
fulladd4 FA1_4(SUM, C_OUT, A, B, C_IN); // Instantiate the 4-bit full adder. call it FA1_4
Initial // Set up the monitoring for the signal values
begin
$monitor($time," A= %b, B=%b, C_IN= %b, --- C_OUT= %b, SUM=%b\n",A, B, C_IN, C_OUT, SUM);
end
Initial  // Stimulate inputs
begin
A = 4'd0; B = 4'd0; C_IN = 1'b0;
#5 A = 4'd3; B = 4'd4;
#5 A = 4'd2; B = 4'd5;
#5 A = 4'd9; B = 4'd9;
#5 A = 4'd10; B = 4'd15;
#5 A = 4'd10; B = 4'd5; C_IN = 1'b1;
end
endmodule
```

Output:

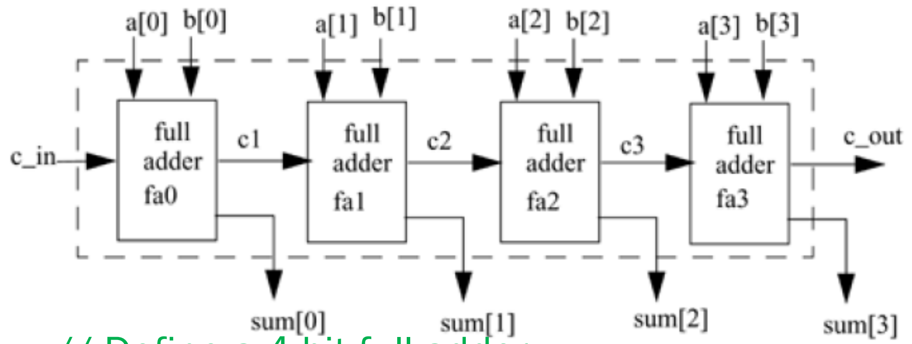
```
0 A= 0000, B=0000, C_IN= 0, --- C_OUT= 0, SUM= 0000
5 A= 0011, B=0100, C_IN= 0, --- C_OUT= 0, SUM= 0111
10 A= 0010, B=0101, C_IN= 0, --- C_OUT= 0, SUM= 0111
15 A= 1001, B=1001, C_IN= 0, --- C_OUT= 1, SUM= 0010
20 A= 1010, B=1111, C_IN= 0, --- C_OUT= 1, SUM= 1001
25 A= 1010, B=0101, C_IN= 1,, C_OUT= 1, SUM= 0000
```

Gate-Level Modeling or Structural Modeling

Verilog Description of 4-bit Ripple Carry full adder design



```
// Define a 1-bit full adder
module fulladd(sum, c_out, a, b, c_in);
// I/O port declarations
output sum, c_out;
input a, b, c_in;
// Internal nets
wire s1, c1, c2;
// Instantiate logic gate primitives
xor (s1, a, b);
and (c1, a, b);
xor (sum, s1, c_in);
and (c2, s1, c_in);
xor (c_out, c2, c1);
endmodule
```



```
// Define a 4-bit full adder
module fulladd4(sum, c_out, a, b, c_in);
// I/O port declarations
output [3:0] sum;
output c_out;
Input [3:0] a, b;
input c_in;
// Internal nets
wire c1, c2, c3;
// Instantiate four 1-bit full adders.
fulladd fa0(sum[0], c1, a[0], b[0], c_in);
fulladd fa1(sum[1], c2, a[1], b[1], c1);
fulladd fa2(sum[2], c3, a[2], b[2], c2);
fulladd fa3(sum[3], c_out, a[3], b[3], c3);
endmodule
```

Gate-Level Modeling or Structural Modeling

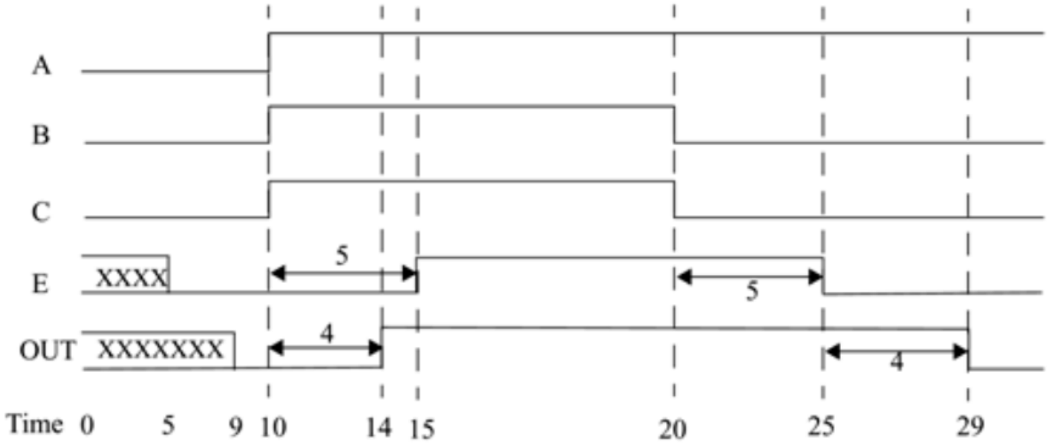
Types of Delay Specification Syntax

```
// Delay of delay_time for all transitions
and #(delay_time) a1(out, i1, i2);
// Rise and Fall Delay Specification.
and #(rise_val, fall_val) a2(out, i1, i2);
// Rise, Fall, and Turn-off Delay Specification
bufif0 #(rise_val, fall_val, turnoff_val) b1 (out, in, control);
```

Examples of delay specification

```
// Define a simple combination module called D
module D (out, a, b, c);
// I/O port declarations
output out;
input a,b,c;
// Internal nets
wire e;
// Instantiate primitive gates to build the circuit
and #(5) a1(e, a, b); //Delay of 5 on gate a1
or #(4) o1(out, e,c); //Delay of 4 on gate o1
endmodule
```

```
// Stimulus (top-level module)
module stimulus;
reg A, B, C;
wire OUT;
// Instantiate the module D
D d1( OUT, A, B, C);
// Stimulate the inputs. Finish the simulation at 40 time units.
initial
begin
A= 1'b0; B= 1'b0; C= 1'b0;
#10 A= 1'b1; B= 1'b1; C= 1'b1;
#10 A= 1'b1; B= 1'b0; C= 1'b0;
#20 $finish;
end
endmodule
```


A
B
C
E
XXXX
5
5
OUT
XXXXXXX
4
4
Time 0
5
9
10
14
15
20
25
29