KYUNGPOOK NATIONAL UNIVERSITY

COMP311-1 Logic Circuit Design Chap. 10

Instructor: Taigon Song (송대건)

2019 Fall

10.1 Types of Delay Models

Q: How would we want to model the delay of modules?

- Delay modeling before considering the actual process-related delay values are very important in Verilog
- Consider the designed module below to perform delay modeling
  - To do this, we can consider three scenarios

딜레이의 3가지 종류

Figure 10-1. Distributed Delay

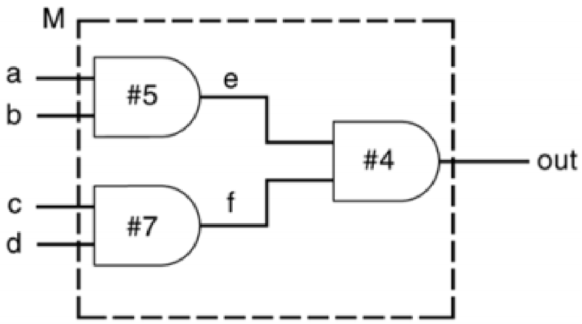


Figure 10-2. Lumped Delay

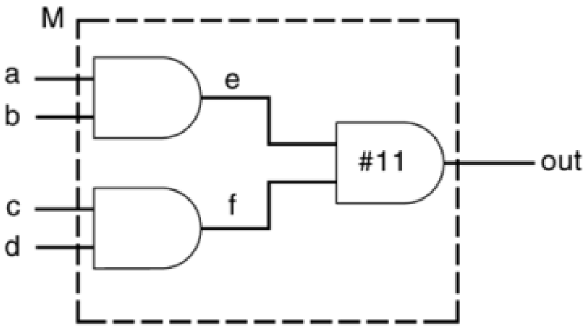
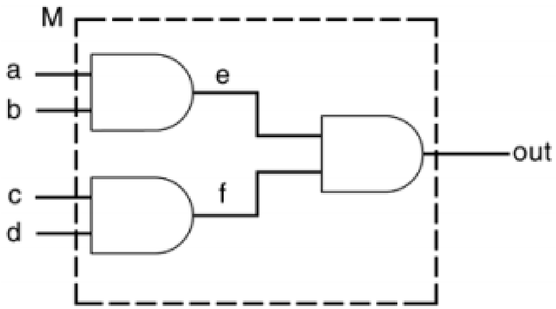


Figure 10-3. Pin-to-Pin Delay



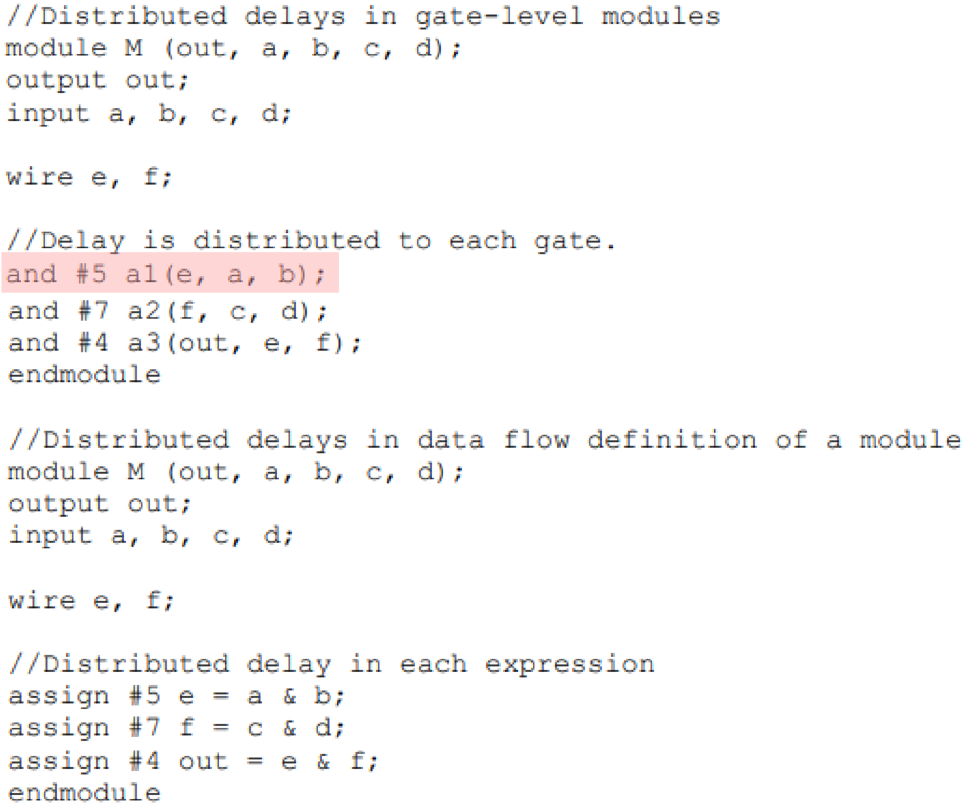
path a—e—out, delay = 9
path b—e—out, delay = 9
path c—f—out, de;ay = 11
path d—f—out, delay = 11

10.1 Types of Delay Models

10.1.1 Distributed Delay

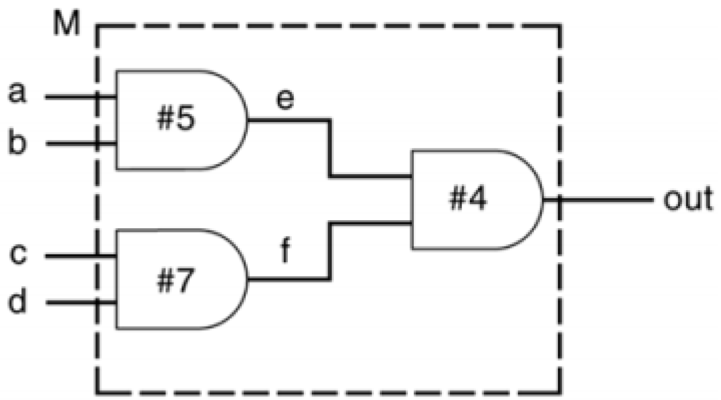
- Delay values that are specified on a per-element basis
  - Delay values are assigned to individual elements in the circuit

Example 10-1 Distributed Delays



```
//Distributed delays in gate-level modules
module M (out, a, b, c, d);
output out;
input a, b, c, d;

wire e, f;

//Delay is distributed to each gate.
and #5 a1(e, a, b);
and #7 a2(f, c, d);
and #4 a3(out, e, f);
endmodule

//Distributed delays in data flow definition of a module
module M (out, a, b, c, d);
output out;
input a, b, c, d;

wire e, f;

//Distributed delay in each expression
assign #5 e = a & b;
assign #7 f = c & d;
assign #4 out = e & f;
endmodule
```

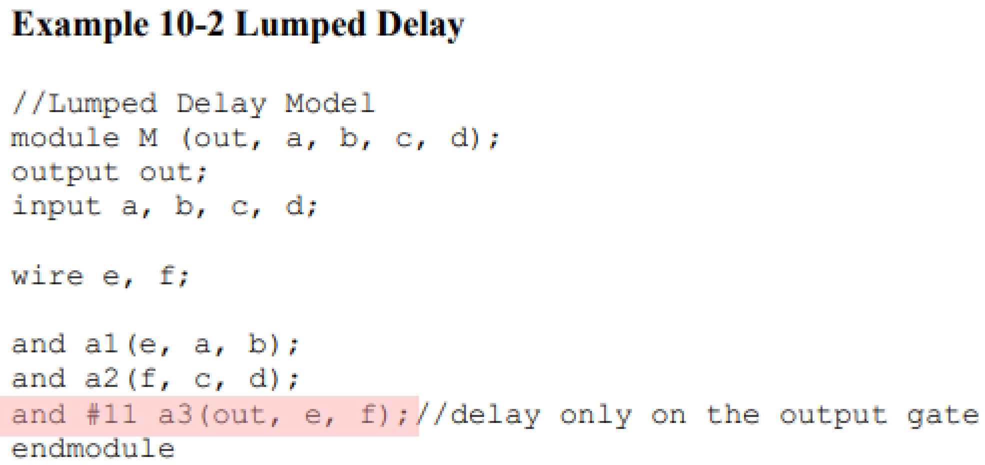
Figure 10-1. Distributed Delay



10.1 Types of Delay Models

10.1.2 Lumped Delay

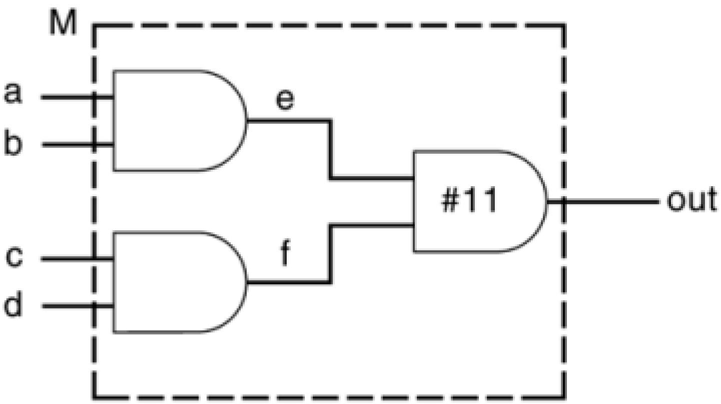
- Delay values that are specified on a per-module basis
  - Cumulative delay lumped at one location



Example 10-2 Lumped Delay

```
//Lumped Delay Model
module M (out, a, b, c, d);
output out;
input a, b, c, d;

wire e, f;

and a1(e, a, b);
and a2(f, c, d);
and #11 a3(out, e, f);//delay only on the output gate
endmodule
```

Figure 10-2. Lumped Delay

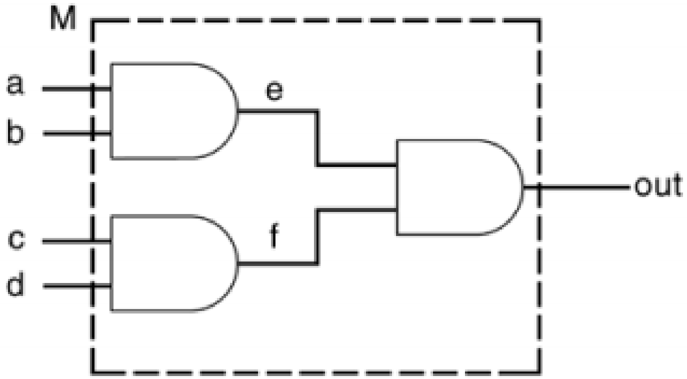


10.1 Types of Delay Models

10.1.3 Pin-to-Pin Delays

- Delay values are assigned individually to paths from each input to each output
    - For large circuits, they are easier to model than distributed delays
        - Designers only need to know the I/O pins rather than the internals of the module
    - Also known as “path delays”

Figure 10-3. Pin-to-Pin Delay

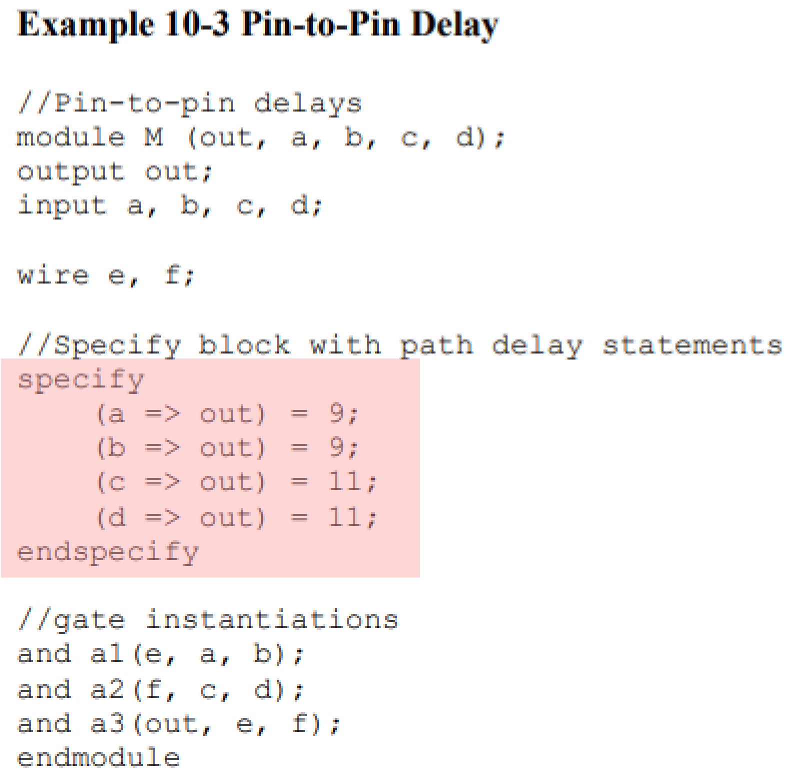


path a—e—out, delay = 9
path b—e—out, delay = 9
path c—f—out, de;ay = 11
path d—f—out, delay = 11

10.2 Path Delay Modeling

10.2.1 Specify Blocks

- Path delay: a delay between a source (input or inout) pin and a destination (output or inout) pin of a module
- Assigned using keywords “specify” and “endspecify”
  - Does not appear under any other block such as initial or always



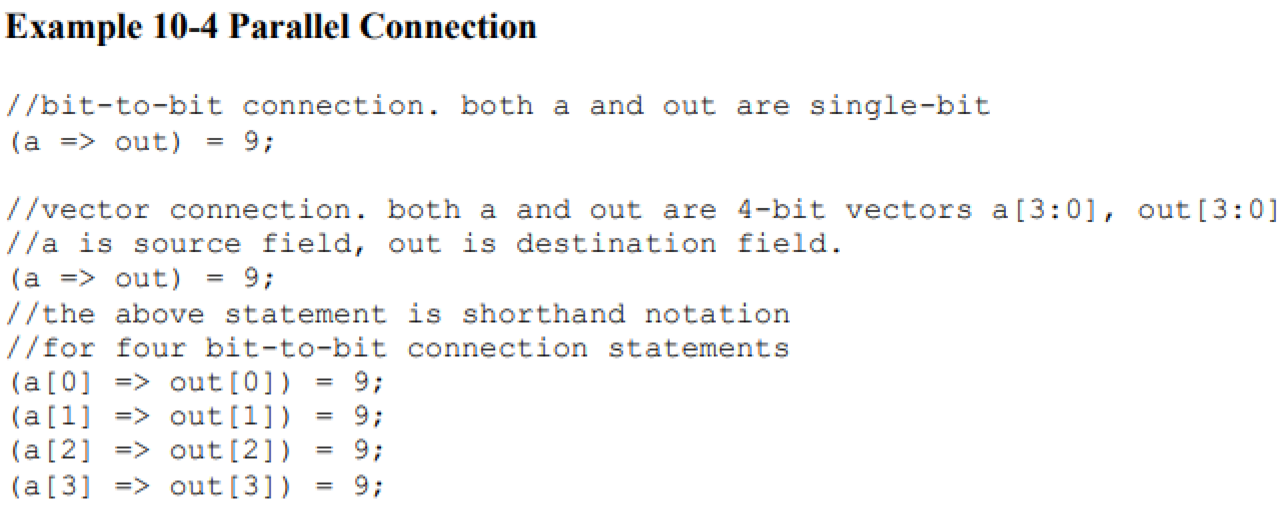
Example 10-3 Pin-to-Pin Delay

```
//Pin-to-pin delays
module M (out, a, b, c, d);
output out;
input a, b, c, d;

wire e, f;

//Specify block with path delay statements
specify
    (a => out) = 9;
    (b => out) = 9;
    (c => out) = 11;
    (d => out) = 11;
endspecify

//gate instantiations
and a1(e, a, b);
and a2(f, c, d);
and a3(out, e, f);
endmodule
```

10.2 Path Delay Modeling

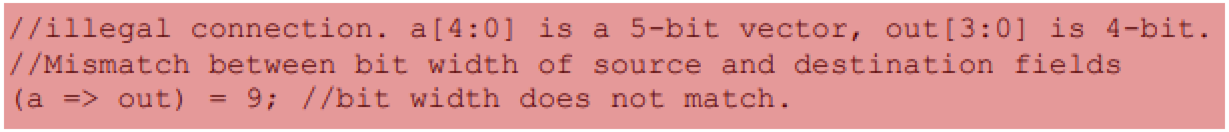
10.2.2 Inside Specify Blocks

- What can be done inside specify blocks
  - Parallel connection
    - Usage: (<source_field> => <destination_field>) = <delay_value>;
    - When source and destination fields are vectors, they must have same # of bits



Example 10-4 Parallel Connection

```
//bit-to-bit connection. both a and out are single-bit
(a => out) = 9;

//vector connection. both a and out are 4-bit vectors a[3:0], out[3:0]
//a is source field, out is destination field.
(a => out) = 9;
//the above statement is shorthand notation
//for four bit-to-bit connection statements
(a[0] => out[0]) = 9;
(a[1] => out[1]) = 9;
(a[2] => out[2]) = 9;
(a[3] => out[3]) = 9;
```

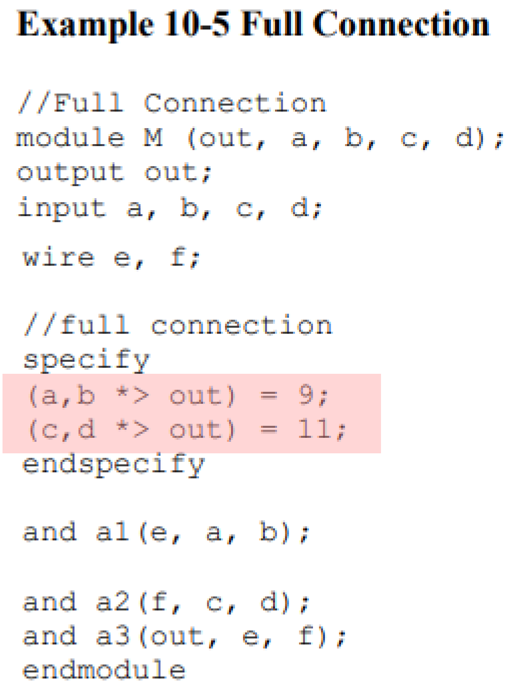


```
//illegal connection. a[4:0] is a 5-bit vector, out[3:0] is 4-bit.
//Mismatch between bit width of source and destination fields
(a => out) = 9; //bit width does not match.
```

10.2 Path Delay Modeling

10.2.2 Inside Specify Blocks

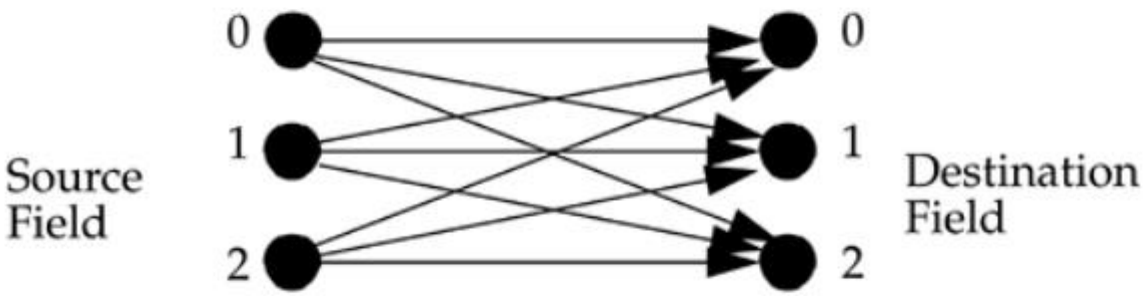
- What can be done inside specify blocks
  - Full connection
    - Usage: (<source_field> *> <destination_field>) = <delay_value>;
    - When source/destination are vectors, they need not have the same # of bits



Example 10-5 Full Connection

```
//Full Connection
module M (out, a, b, c, d);
output out;
input a, b, c, d;
wire e, f;

//full connection
specify
(a,b *> out) = 9;
(c,d *> out) = 11;
endspecify

and a1(e, a, b);

and a2(f, c, d);
and a3(out, e, f);
endmodule
```

Figure 10-5. Full Connection

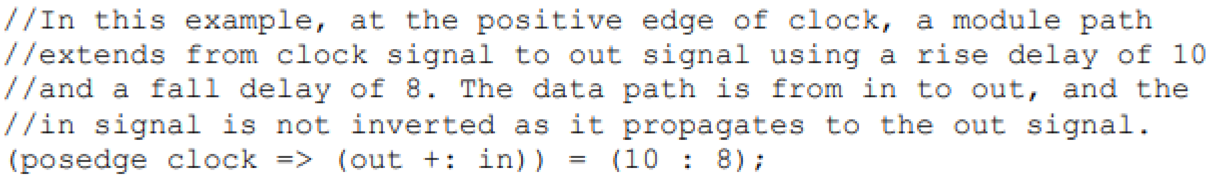


```
//a[31:0] is a 32-bit vector and out[15:0] is a 16-bit vector
//Delay of 9 between each bit of a and every bit of out

specify
( a *> out) = 9; // you would need 32 X 16 = 352 parallel connection
                // statements to accomplish the same result! Why?
endspecify
```

10.2 Path Delay Modeling

10.2.2 Inside Specify Blocks

- What can be done inside specify blocks
  - Edge-Sensitive Paths
    - Example below: a path from in to out
    - Rise delay: 10, fall delay: 8 when at posedge of clock



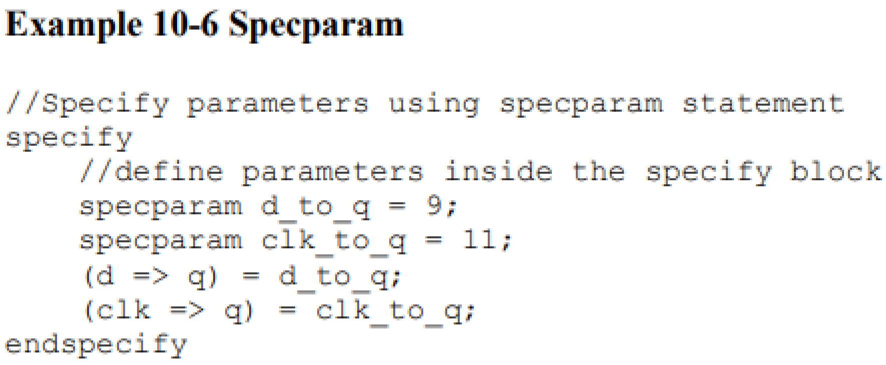
```
//In this example, at the positive edge of clock, a module path
//extends from clock signal to out signal using a rise delay of 10
//and a fall delay of 8. The data path is from in to out, and the
//in signal is not inverted as it propagates to the out signal.
(posedge clock => (out +: in)) = (10 : 8);
```

10.2 Path Delay Modeling

10.2.2 Inside Specify Blocks

변수를 가급적 위에 두라
변수선언시 숫자를 바로집어넣지말고 다른변수로 재정의
변수선언시 앞에서 할수있는건 다시 정의해놓기

- What can be done inside specify blocks
  - specparam statements
    - Provided for convenience in assigning delays
    - Recommended to provide ALL pin-to-pin delay values in specify parameters rather than hard-coded numbers



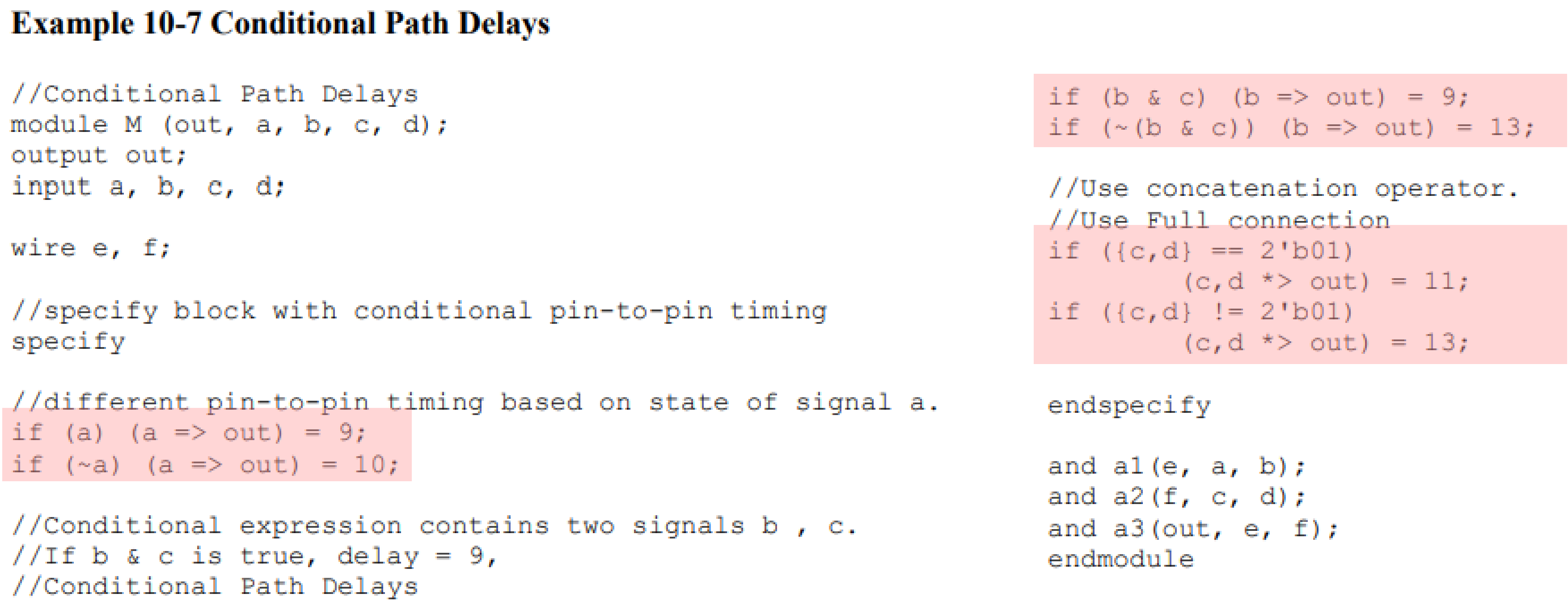
Example 10-6 Specparam

```
//Specify parameters using specparam statement
specify
    //define parameters inside the specify block
    specparam d_to_q = 9;
    specparam clk_to_q = 11;
    (d => q) = d_to_q;
    (clk => q) = clk_to_q;
endspecify
```

10.2 Path Delay Modeling

10.2.2 Inside Specify Blocks

- What can be done inside specify blocks
  - Conditional path delays
    - Also known as state dependent path delays (SDPD)



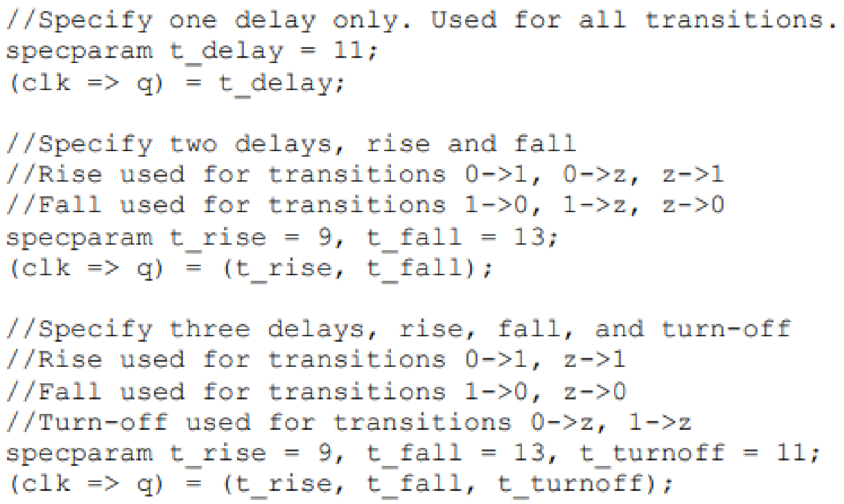
Example 10-7 Conditional Path Delays

```
//Conditional Path Delays
module M (out, a, b, c, d);
output out;
input a, b, c, d;

wire e, f;

//specify block with conditional pin-to-pin timing
specify

//different pin-to-pin timing based on state of signal a.
if (a) (a => out) = 9;
if (~a) (a => out) = 10;

//Conditional expression contains two signals b , c.
//If b & c is true, delay = 9,
//Conditional Path Delays
if (b & c) (b => out) = 9;
if (~(b & c)) (b => out) = 13;

//Use concatenation operator.
//Use Full connection
if ({c,d} == 2'b01)
        (c,d *> out) = 11;
if ({c,d} != 2'b01)
        (c,d *> out) = 13;

endspecify

and a1(e, a, b);
and a2(f, c, d);
and a3(out, e, f);
endmodule
```

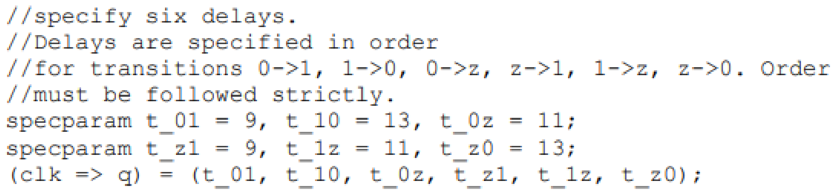
10.2 Path Delay Modeling

10.2.2 Inside Specify Blocks

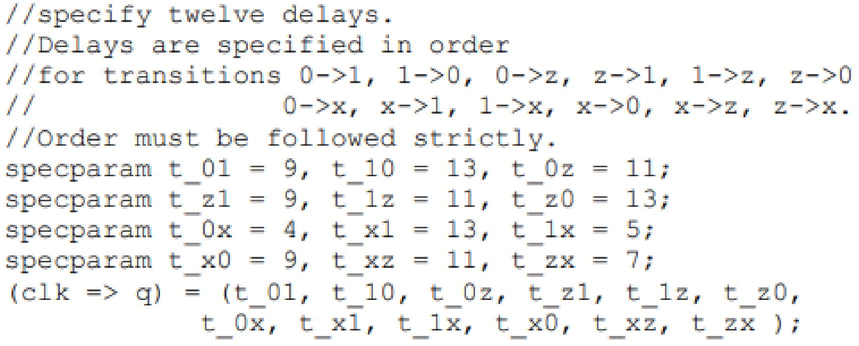
- What can be done inside specify blocks
  - Rise, fall, and turn-off delays
  - To be defined only as one/two/three/six/twelve delay values



```
//Specify one delay only. Used for all transitions.
specparam t_delay = 11;
(clk => q) = t_delay;

//Specify two delays, rise and fall
//Rise used for transitions 0->1, 0->z, z->1
//Fall used for transitions 1->0, 1->z, z->0
specparam t_rise = 9, t_fall = 13;
(clk => q) = (t_rise, t_fall);

//Specify three delays, rise, fall, and turn-off
//Rise used for transitions 0->1, z->1
//Fall used for transitions 1->0, z->0
//Turn-off used for transitions 0->z, 1->z
specparam t_rise = 9, t_fall = 13, t_turnoff = 11;
(clk => q) = (t_rise, t_fall, t_turnoff);
```



```
//specify six delays.
//Delays are specified in order
//for transitions 0->1, 1->0, 0->z, z->1, 1->z, z->0. Order
//must be followed strictly.
specparam t_01 = 9, t_10 = 13, t_0z = 11;
specparam t_z1 = 9, t_1z = 11, t_z0 = 13;
(clk => q) = (t_01, t_10, t_0z, t_z1, t_1z, t_z0);
```

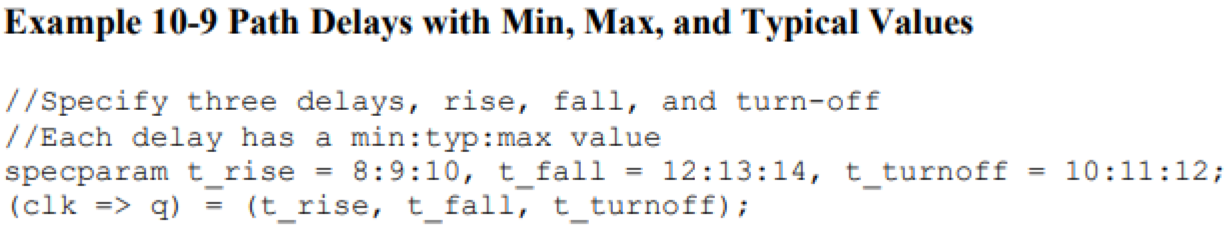


```
//specify twelve delays.
//Delays are specified in order
//for transitions 0->1, 1->0, 0->z, z->1, 1->z, z->0
//                0->x, x->1, 1->x, x->0, x->z, z->x.
//Order must be followed strictly.
specparam t_01 = 9, t_10 = 13, t_0z = 11;
specparam t_z1 = 9, t_1z = 11, t_z0 = 13;
specparam t_0x = 4, t_x1 = 13, t_1x = 5;
specparam t_x0 = 9, t_xz = 11, t_zx = 7;
(clk => q) = (t_01, t_10, t_0z, t_z1, t_1z, t_z0,
              t_0x, t_x1, t_1x, t_x0, t_xz, t_zx );
```

10.2 Path Delay Modeling

10.2.2 Inside Specify Blocks

- What can be done inside specify blocks
  - Min, max, and typical delays
    - Can be combined with various delay-related statements



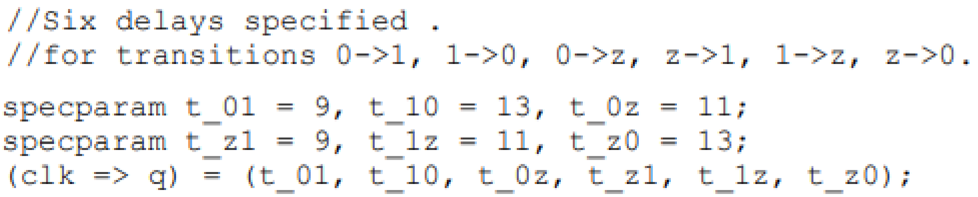
Example 10-9 Path Delays with Min, Max, and Typical Values

```
//Specify three delays, rise, fall, and turn-off
//Each delay has a min:typ:max value
specparam t_rise = 8:9:10, t_fall = 12:13:14, t_turnoff = 10:11:12;
(clk => q) = (t_rise, t_fall, t_turnoff);
```

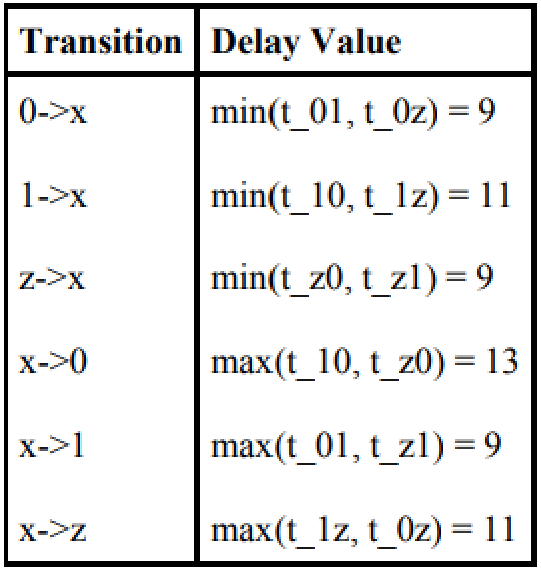
10.2 Path Delay Modeling

10.2.2 Inside Specify Blocks

- What can be done inside specify blocks
  - Handling x transitions
    - When ‘x’, transition delays are not explicitly specified,
      - Transitions from ‘x’ to a known state → maximum possible time
      - Transitions from a known state to ‘x’ → minimum possible time



```
//Six delays specified .
//for transitions 0->1, 1->0, 0->z, z->1, 1->z, z->0.
specparam t_01 = 9, t_10 = 13, t_0z = 11;
specparam t_z1 = 9, t_1z = 11, t_z0 = 13;
(clk => q) = (t_01, t_10, t_0z, t_z1, t_1z, t_z0);
```

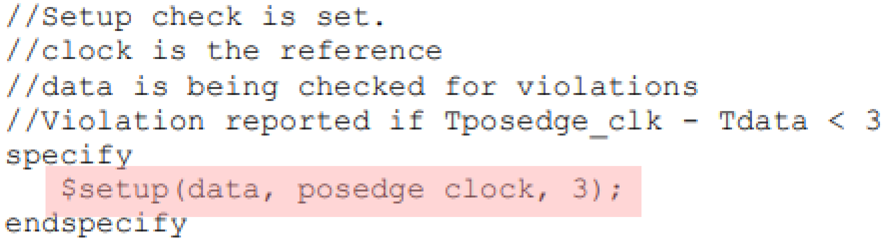


Transition	Delay Value
0->x	min(t_01, t_0z) = 9
1->x	min(t_10, t_1z) = 11
z->x	min(t_z0, t_z1) = 9
x->0	max(t_10, t_z0) = 13
x->1	max(t_01, t_z1) = 9
x->z	max(t_1z, t_0z) = 11

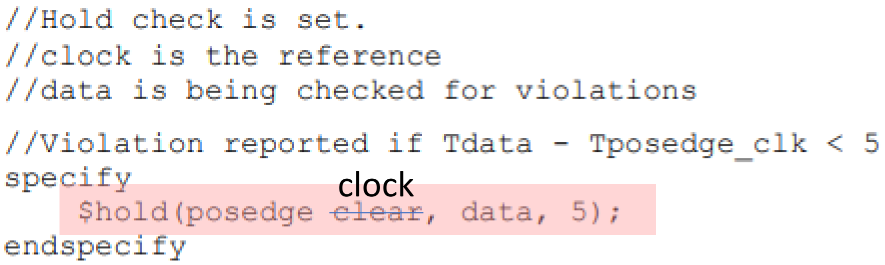
10.3 Timing Checks

skip

10.3.1 $setup and $hold Checks

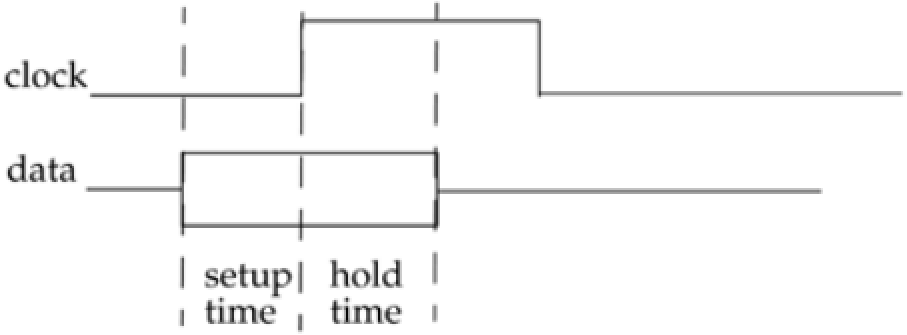


```
//Setup check is set.
//clock is the reference
//data is being checked for violations
//Violation reported if Tposedge_clk - Tdata < 3
specify
   $setup(data, posedge clock, 3);
endspecify
```



```
//Hold check is set.
//clock is the reference
//data is being checked for violations

//Violation reported if Tdata - Tposedge_clk < 5
specify
    $hold(posedge ~~clear~~ clock, data, 5);
endspecify
```

Figure 10-6. Setup and Hold Times

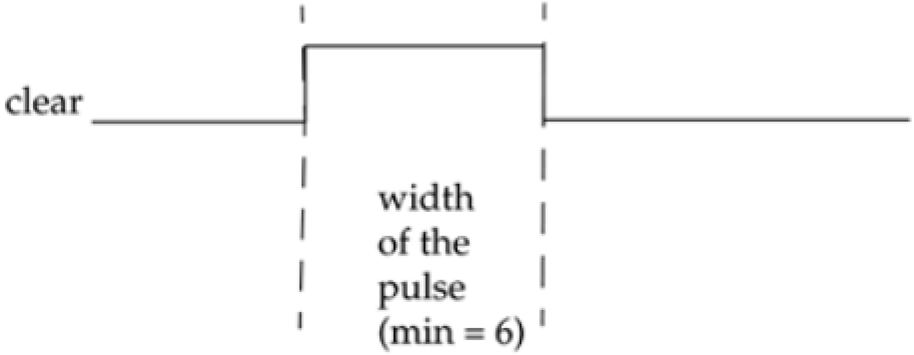


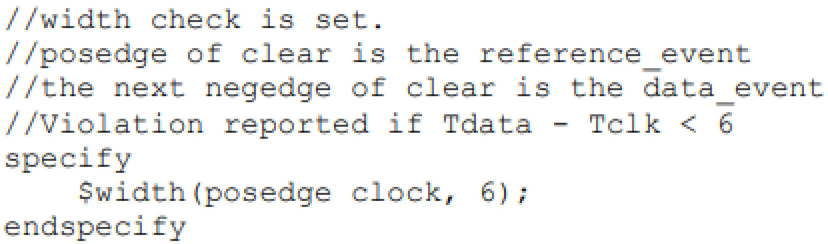
10.3 Timing Checks

10.3.2 $width Check

skip

- Checks the width of pulse
  - Useful for checking asynchronous input widths





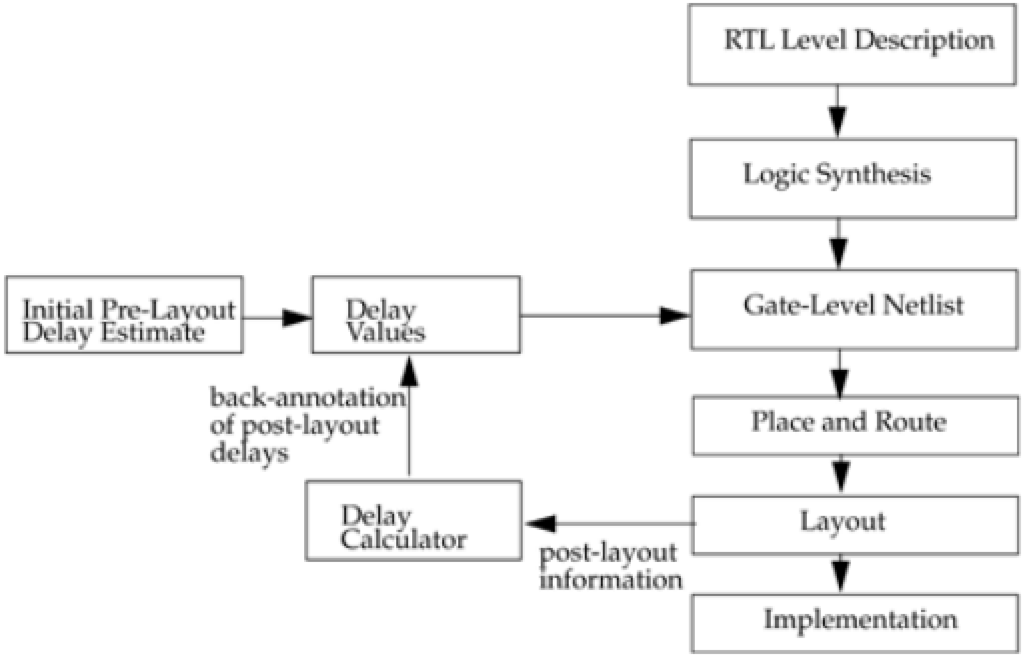
```
//width check is set.
//posedge of clear is the reference_event
//the next negedge of clear is the data_event
//Violation reported if Tdata - Tclk < 6
specify
    $width(posedge clock, 6);
endspecify
```

10.4 Delay Back-Annotation

skip

- A process of updating the pre-layout delays in the chip by actual fabrication process delay values
  - May not be too important these days depending on the technology libraries and design steps/goals

Figure 10-7. Delay Back-Annotation



In-class Assignment

Design a 16-bit x 128 Memory (SRAM)

- I/O ports
  - input [15:0] datamem_data, datamem_addr…?,
  - input clk, rst, we, re,
  - output [15:0] datamem_strm

- Function
  - Writes input to the memory when ‘we’
  - Reads data to the output when ‘re’