GATE DELAY

- The signal propagation delay from any gate input to the gate output can be specified using a gate delay.
- The gate delay can be specified in the gate instantiation itself.
- Here is the syntax of a gate instantiation with the delay specification

```
gate_type [ delay ] [ instance_name ] ( terminal_list );
```

- The delay specifies the gate delay, that is, the propagation delay from any gate input to the output. When no gate delay is specified, the default delay is zero.

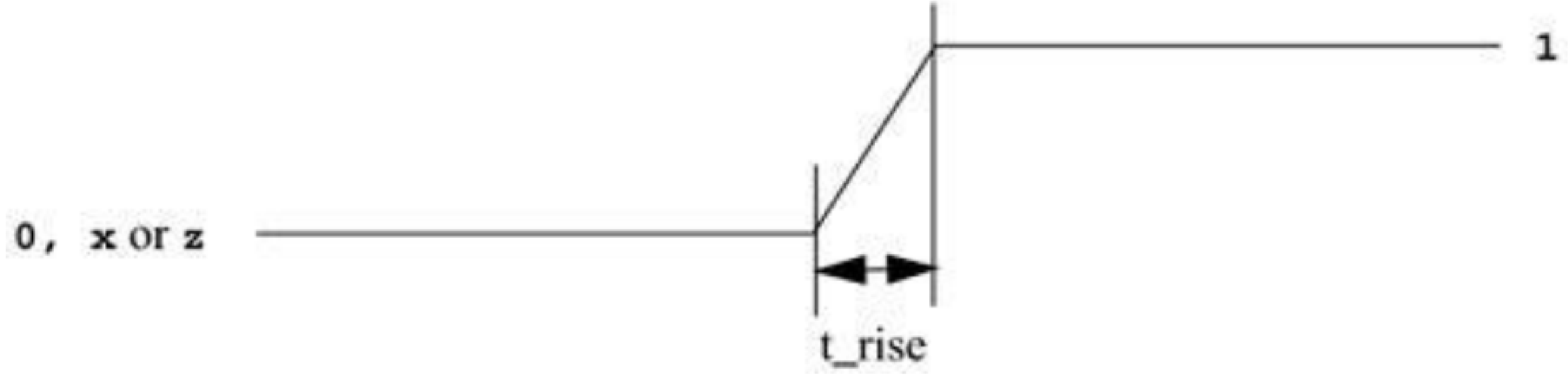
- A gate delay can be comprised of up to three values:

i. rise delay
ii. fall delay
iii. turn-off delay

A delay specification may contain zero, one, two, or all three values specified.

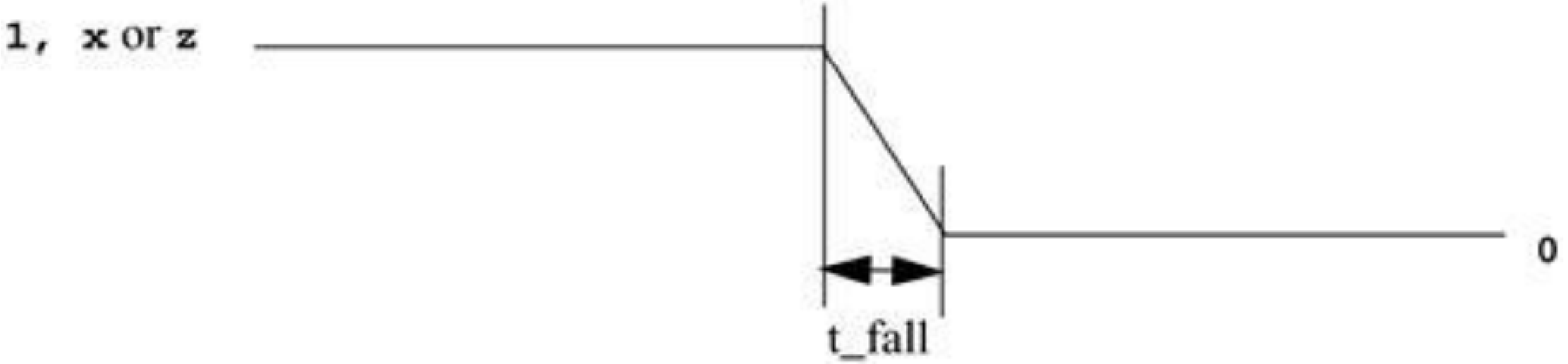
Rise delay

The rise delay is associated with a gate output transition to a 1 from another value.



Fall delay

The fall delay is associated with a gate output transition to a 0 from another value.



Turn-off delay

- The turn-off delay is associated with a gate output transition to the high impedance value (z) from another value. If the value changes to x, the minimum of the three delays is considered.

Note:

- If only one delay is specified, this value is used for all transitions.
- If two delays are specified, they refer to the rise and fall delay values. The turn-off delay is the minimum of the two delays.
- If all three delays are specified, they refer to rise, fall, and turn-off delay values.
- If no delays are specified, the default value is zero.

- The following table shows the values that are used for a delay based on the number of specified values

	No delay	1 value (d)	2 values (d1, d2)	3 values (dA, dB, dC)
Rise	0	d	d1	dA
Fall	0	d	d2	dB
To_x	0	d	min[a] (d1, d2)	min (dA, dB, dC)
Turn-off	0	d	min (d1, d2)	dC

a. min: minimum

- Notice that the transition to x delay(to_x) cannot be explicitly specified but is determined from the other specified values.
- Note that all delays in a Verilog HDL model are expressed in terms of time units. The association of time units with actual time is done using the time scale compiler directive.
- In the following instantiation, the gate delay is 0 since no delay has been specified

```
not N1 (Qbar, Q);
```

- In the gate instantiation,

```
nand #6 (Out, In1, In2);
```

all delays are 6, that is, the rise delay and fall delay are both 6. Turn-off delay does not apply to a nand gate since the output never goes into high impedance. The transition to x delay is also 6.

```
and #(3, 5) (Out, In1, In2, In3);
```

- In this gate instantiation, the rise delay has been specified to be 3, the fall delay is 5 and the transition to x delay is the minimum of 3 and 5, which is 3.

```
notif1 #(2, 8, 6) (Dout, Din1, Din2);
```

- In the above gate instance the rise delay is 2, the fall delay is 8, the turn-off delay is 6 and the transition to x delay is the minimum of 2, 8 and 6, which is 2.
- Multiple-input gates, such as and and or, and multiple-output gates (buf and not) can have only up to two delays specified (since output never goes to z).
- Tristate gates can have up to three delays and the pull gates cannot have any delays

Min : typ: max Delay Form

- Verilog provides an additional level of control for each type of delay mentioned above. For each type of delay(rise, fall, and turn-off) three values, min, typ, and max, can be specified.
- Min/typ/max values are used to model devices whose delays vary within a minimum and maximum range because of the IC fabrication process variations.
- The selection of which delay to use is usually made as an option during a simulation run.
- For example, if maximum delay simulation is performed, a rise delay of 4 and a fall delay of 7 is used for the nand gate instance.

Min value

The min value is the minimum delay value that the designer expects the gate to have.

Typ val

The typ value is the typical delay value that the designer expects the gate to have.

Max value

The max value is the maximum delay value that the designer expects the gate to have.

Method of choosing a min/typ/max value may vary for different simulators or operating systems. (For VerilogXL , the values are chosen by specifying options +maxdelays, +typdelays, and +mindelays at run time.

Form

- A delay for a gate (including all other delays such as in continuous assignments) can also be specified in a min:typ:max form.
- The form is:

```
minimum : typical : maximum
```

- The minimum, typical and maximum values must be constant expressions.

Example: A delay in this form used in a gate instantiation.

```
nand #(2:3:4, 5:6:7) (Pout, Pin1, Pin2);
```

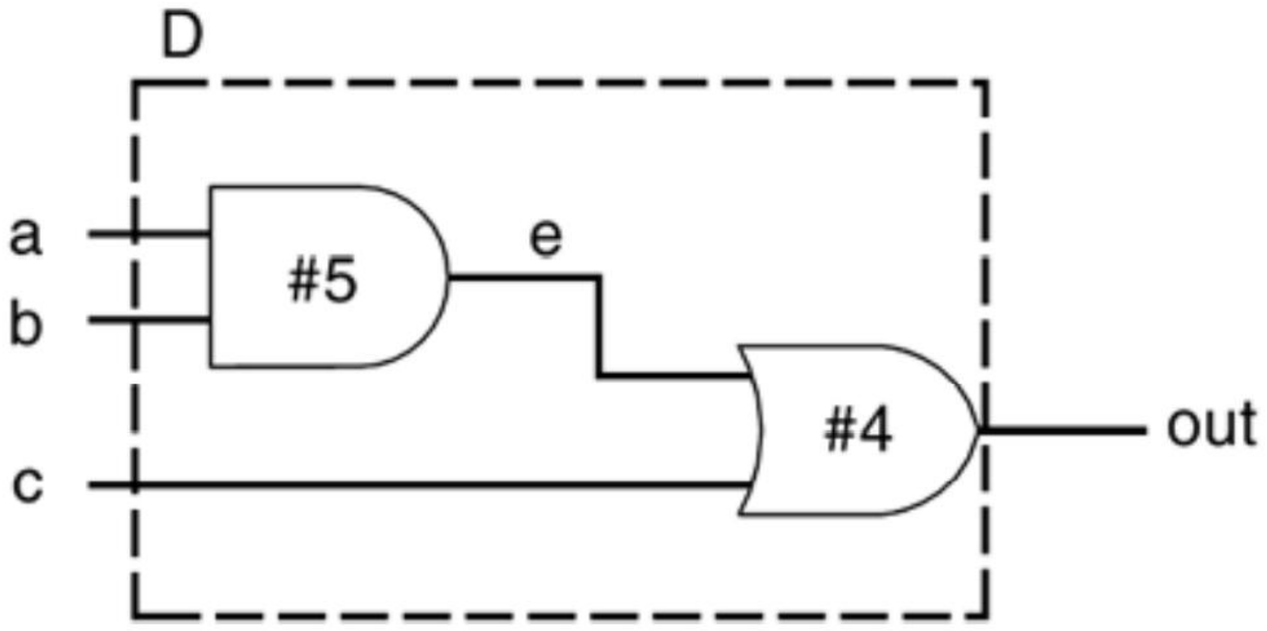
Examples of min, typ, and max value specification for Verilog-XL are shown

```
// One delay
// if +mindelays, delay= 4
// if +typdelays, delay= 5
// if +maxdelays, delay= 6
and #(4:5:6) a1(out, i1, i2);

// Two delays
// if +mindelays, rise= 3, fall= 5, turn-off = min(3,5)
// if +typdelays, rise= 4, fall= 6, turn-off = min(4,6)
// if +maxdelays, rise= 5, fall= 7, turn-off = min(5,7)
and #(3:4:5, 5:6:7) a2(out, i1, i2);

// Three delays
// if +mindelays, rise= 2 fall= 3 turn-off = 4
// if +typdelays, rise= 3 fall= 4 turn-off = 5
// if +maxdelays, rise= 4 fall= 5 turn-off = 6
and #(2:3:4, 3:4:5, 4:5:6) a3(out, i1,i2);
```

Delay Example

- A simple module called D implements the following logic equations: **out = (a b) + c**

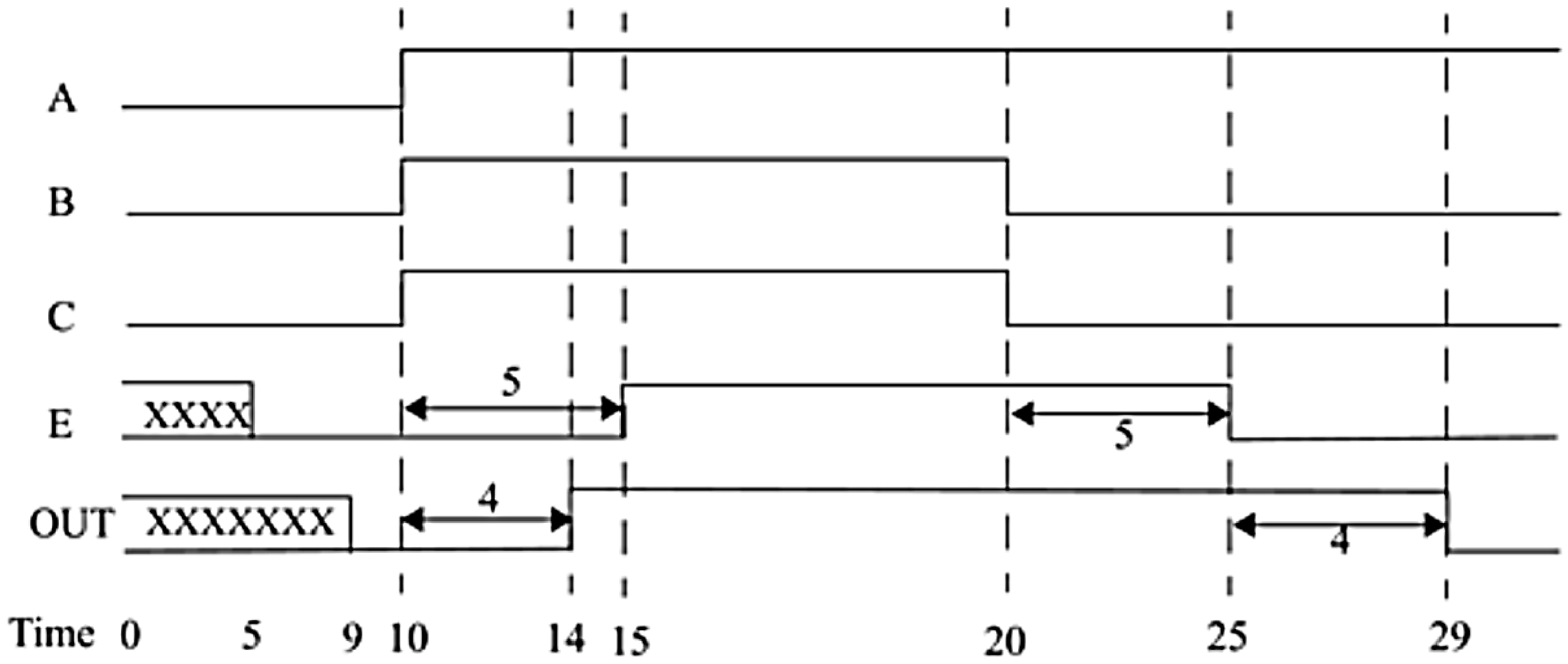
- The module contains two gates with delays of 5 and 4 time units



Verilog Definition for Module D with Delay

```
// 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
```

Waveforms for Delay Simulation



Array of Instances

- There are many situations when repetitive instances are required.
- These instances differ from each other only by the index of the vector to which they are connected.
- To simplify specification of such instances, Verilog HDL allows an array of primitive instances to be defined
- When repetitive instances are required, a range specification can optionally be specified in a gate instantiation (a range specification can also be used in a module instantiation).

Array of Instances

- The syntax of a gate instantiation in this case is:

```
gate_type [ delay ] instance_name [ leftbound : rightbound ]
        ( list _of_ terminal_names );
```

- The left bound and right bound values are any two constant expressions.
- It is not necessary for the left bound to be greater than the right bound and either of the bounds is not restricted to be a 0.

Here is an example

```
wire [3:0] Out, InA, InB;
. . .
nand Gang [3 : 0] (Out, InA, InB);
```

This instantiation with the range specification is same as:

```
nand
   Gang3 (Out[3], InA[3], InB[3]),
   Gang2 (Out[2], InA[2], InB[2]),
   Gang1 (Out[1], InA[1], InB[1]),
   Gang0 (Out[0], InA[0], InB[0]);
```

Note that the instance name is not optional when specifying an array of instances

Example 5-4 Simple Array of Primitive Instances

```
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]);
```

Implicit Nets

- If a net is not declared in a Verilog HDL model, by default, it is implicitly declared as a 1-bitwire.
- However the **\"default_nettype** compiler directive can be used to override the default net type.
- This compiler directive is of the form:

```
`default_nettype net_type
```

Here is an example.

```
`default_nettype wand
```

- With this directive, all subsequent undeclared nets are of type wand.
- The \"default_nettype compiler directive occurs out side of a module definition and stays in effect until the next same directive is reached or a reset all directive is found

Parameters

- Verilog allows constants to be defined in a module by the keyword parameter
- It is often used to specify delays and widths of variables.
- A parameter can be assigned a value only once, using a parameter declaration.
- Parameters cannot be used as variables.
- Parameter values for each module instance can be overridden individually at compile time. This allows the module instances to be customized. Parameter types and sizes can also be defined

Parameter declaration form:

```
parameter param1 = const_expr1 , param2 = const_exp2 , . . . ,
          paramN = const_exprN ;
```

Here are some examples.

```
parameter LINELENGTH = 132, ALL_X_S = 16'bx;
parameter BIT = 1, BYTE = 8, PI = 3.14;
parameter STROBE_DELAY = {BYTE + BIT) I 2;
parameter TQ_FILE = \"/home/bhasker/TEST/add.tq\";
```

```
parameter port_id = 5; // Defines a constant port_id
parameter cache_line_width = 256; // Constant defines width of cache
line
parameter signed [15:0] WIDTH; // Fixed sign and range for parameter
                               // WIDTH
```

- Module definitions may be written in terms of parameters.
- Hardcoded numbers should be avoided.
- A parameter value can also be changed at compile time. This is done by using a **defparam** statement or by specifying the parameter value in the module instantiation statement.
- Thus, the use of parameters makes the module definition flexible.
- Module behavior can be altered simply by changing the value of a parameter

NOTE:

- Verilog HDL local parameters (defined using keyword localparam) are identical to parameters except that they cannot be directly modified with the defparam statement or by the ordered or named parameter value assignment.
- The localparam keyword is used to define parameters when their values should not be changed.
- For example, the state encoding for a state machine can be defined using localparam. The state encoding cannot be changed. This provides protection against inadvertent parameter redefinition.

```
localparam state1 = 4'b0001,
           state2 = 4'b0010,
           state3 = 4'b0100,
           state4 = 4'b1000;
```

THANK YOU