Delay Modeling

Dr. Shubhajit Roy Chowdhury,

Center for VLSI and Embedded System Technologies,
IIIT Hyderabad
Outline

- Types of Delays
- Delays in Gate Level Model
- Delays in Data Flow Model
- Advanced Delay Model
- Timing Checks
Introduction

- Delays are crucial in \textit{REAL} simulations
  - Post-synthesis simulation
  - Post-layout simulation

- Delay Models
  - Represent different physical concepts
  - Two types
    - Inertial delay
    - Transport delay
Types of Delays

- Inertial Delay
- Transport Delay
Inertial Delay

- The inertia of a circuit node to change value
- Abstractly models the RC circuit seen at the node
- Different types
  - Input inertial delay
  - Output inertial delay
Transport Delay

- Represents the propagation time of signals from module inputs to its outputs
- Models the internal propagation delays of electrical elements
Delays in Gate Level Model

- Rise Delay
- Fall Delay
- Turn-Off Delay
- Min/Typ/Max Delay Values
Delays in Gate Level Modeling

Delay are shown by # sign in all verilog modeling levels

- Inertial rise delay
- Inertial fall delay
- Inertial turn-off delay

and #(rise_val, fall_val, turnoff_val) a(out, in1, in2)
Gate Delays
Rise, Fall, Turn-off delays

- $0, x, \text{or } z \rightarrow 1$
  - $t_{\text{rise}}$

- $1, x, \text{or } z \rightarrow 0$
  - $t_{\text{fall}}$

- $0, 1 \text{ or } x \rightarrow z$
  - $t_{\text{turnoff}}$
Delays in Gate Level Modeling (cont’d)

```verilog
and a0(out,i1,i2);
and #(5) a1(out,i1,i2);
and #(4,6) a2(out,i1,i2);
and #(3,4,5) a3(out,i1,i2);
```

- If no delay specified
  - Default value is zero
- If only one value specified
  - It is used for all three delays
- If two values specified
  - They refer respectively to rise and fall delays
  - Turn-off delay is the minimum of the two
Example: Two delays are specified

and #(3, 2) (out, in1, in2);
Example: Three delays are specified

```
bufif1 #(3, 4, 7) (out, in ctrl);
```

Diagram showing time-domain signals for `in`, `ctrl`, and `out` with specified delays.
Delays in Gate Level Modeling (cont’d)

Min/Typ/Max Values

- Another level of delay control in Verilog
- Each of rise/fall/turnoff delays can have min/typ/max values

\[
\text{not } \#(\text{min:typ:max, min:typ:max, min:typ:max}) \ n(out,in)
\]

- Only one of Min/Typ/Max values can be used in the entire simulation run
  - It is specified at start of simulation, and depends to the simulator used
  - Typ delay is the default
A simple module D implements the logic equations:

\[ \text{out} = (a \cdot b) + c \]
Delays in Data Flow Model
See the difference between c and d

```verilog
wire a, b, d;
wire #2 c;
assign c = a & b;
assign d = a & b;
```
Data Flow Modeling (Contd..)

Example:

```
`timescale 1ns/100ps
module HalfAdder (A, B, Sum, Carry);
input A, B;
output Sum, Carry;
assign #3 Sum = A ^ B;
assign #6 Carry = A & B;
endmodule
```
Data Flow Modeling (Contd..)
Advanced Delay Models

- Distributed Delay Model
- Lumped Delay Model
- Pin-to-Pin (path) Delay Model
Distributed Delay Model

- Delays that are specified on a *per element basis*

- Distributed delays
  - modelled by assigning delay values - in gate level modeling
  - modelled by assigning delays in the continuous assignment - in data flow modeling

- Provides detailed delay modeling
module M (out,a,b,c,d);
output out;
input a,b,c,d;
wire e,f;
and #5 a1(e,a,b);
and #7 a2(f,c,d);
and #4 a3(out,e,f);
endmodule

module M(out,a,b,c,d);
output out;
input a,b,c,d;
wire e,f;
assign #5 e = a & b;
assign #7 f = c & d;
assign #4 out = e & f;
endmodule
Lumped Delay Model

- Lumped delays are specified on a *per module basis*.

- Single delay on the output gate of the module – cumulative delays of all paths is lumped at one location.

- They are easy to model compared with distributed delays
Lumped Delay : Example

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);
endmodule

module M(out,a,b,c,d);
output out;
input a,b,c,d;
wire e,f;
assign e = a & b;
assign f = c & d;
assign #11 out = e & f;
endmodule
Pin-to-Pin Delay Model

- Delays are assigned individually to paths from each input to each output.

- Delays can be separately specified for each input/output path.
Pin-to-Pin Delay: Example

- path a-e-out, delay = 9
- path b-e-out, delay = 9
- path c-f-out, delay = 11
- path d-f-out, delay = 11
Path Delays

- Pin-to-Pin delays are named as path delays
- The delay values got directly from Data Books for standard elements.
- For larger Digital Circuits, a low level circuit simulator like SPICE may be used.
- Designer needs to know the I/O pins of the module rather than the internals of the module – so easier to model, even though it is very detailed.
module M (out,a,b,c,d);
output out;
input a,b,c,d;
wire e,f;

specify
(a => out) = 9;
(b => out) = 9;
(c => out) = 11;
(d => out) = 11;
endspecify

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

- Setup Time
- Hold Time
Introduction to Timing Checks

- In this section, we describe how to set up timing checks to see if any timing constraints are violated during simulation.

- Timing verification is particularly important for timing critical, high speed sequential circuits like microprocessors.
Types of Timing Checks

- System tasks are provided to do timing checks in Verilog.

- There are many timing check system tasks available in Verilog. We will discuss the three most common timing checks tasks:
  - $setup,
  - $hold, and
  - $width.

- All timing checks must be inside specify block.
$setup$ and $hold$ checks

- $setup$ and $hold$ tasks are used to check the setup and hold constraints for a sequential element in the design.
- In a sequential element like an edge-triggered flip-flop, the setup time is the minimum time the data must arrive before the active clock edge.
- The hold time is the minimum time the data cannot change after the active clock edge.

![Diagram of clock and data with setup and hold times]
$setup$ task

- Setup checks can be specified with the system task $setup$.

- Usage: $setup$(data_event, reference_event, limit);
  - data_event: Signal that is monitored for violations
  - reference_event: Signal that establishes a reference for monitoring the data_event signal
  - limit: Minimum time required for setup of data event

- Violation is reported if
  \[(T_{\text{reference_event}} - T_{\text{data_event}}) < \text{limit}\]
$setup example

//Setup check is set.
//Clock is the reference
//Data is being checked for violations
//Violation reported if \( T_{\text{posedge\_clock}} - T_{\text{data}} < 3 \)

specify
$setup(data, posedge\_clock, 3);
endspecify
$hold task

- Hold checks can be specified with the system task $hold.

- Usage: $hold (reference-event, data-event, limit);
  - reference-event: Signal that establishes a reference for monitoring the data-event signal
  - data_event: Signal that is monitored for violation
  - limit: Minimum time required for hold of data event

- Violation is reported if
  \[(T_{\text{data-event}} - T_{\text{reference-event}}) < \text{limit}\]
$hold example

//Hold check is set.
//Clock is the reference
//Data is being checked for violations
//Violation reported if \( T_{\text{data}} - T_{\text{posedge-clk}} < 5 \)

specify
$hold(posedge clear, data, 5);
endspecify
$width$ check

- Sometimes it is necessary to check the width of a pulse.

- The system task $width$ is used to check whether the width of a pulse meets the minimum width requirement.
$width$ check

- **Usage:** $width$(reference-event, limit);
  - **reference-event**: Edge-triggered event (edge transition of a signal)
  - **limit**: Minimum width of the pulse
  - The data_event is not specified explicitly for $width$ but is derived as the next opposite edge of the reference_event signal.

- Violation is reported if
  \[(T_{data-event} - T_{reference-event}) < limit\]
$width example

// Width check is set.
// Posedge of clear is the reference-event
// The next negedge of clear is the data-event
// Violation reported if $T_{data} - T_{Clk} < 6

specify
$width (posedge clock, 6 ) ;
endspecify