

Modeling Approaches in Verilog

Sukanta Pramanik

(Modified from Winter 11 slides by Omid Ardakanian)

CS 450/650 - Computer Architecture

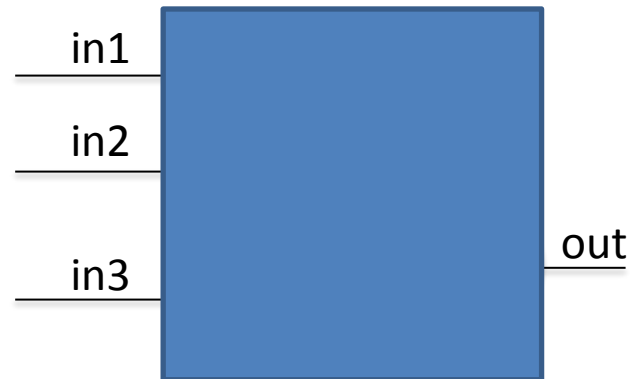
University of Waterloo

Modeling Approaches

- Modeling approaches
 - Structural (gate-level) low level of abstraction
 - Dataflow
 - Behavioural high level of abstraction

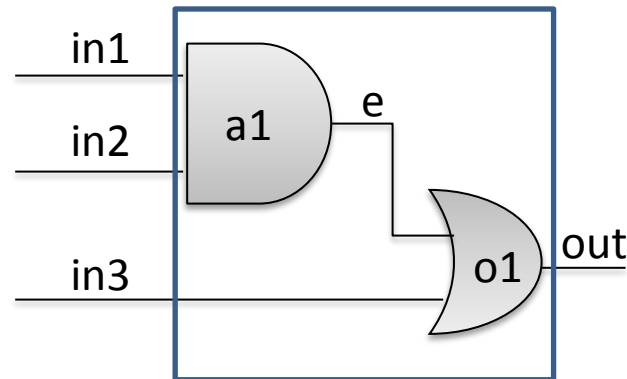
Gate-level Modeling

- Encapsulated Box
- Communicates Through Ports
- Example:



Gate-level Modeling

- Instantiation
 - Gate primitives
 - Defined modules
- Connectivity
- Example:
 - and a1 (e,in1,in2)
 - or o1 (out,in3,e)



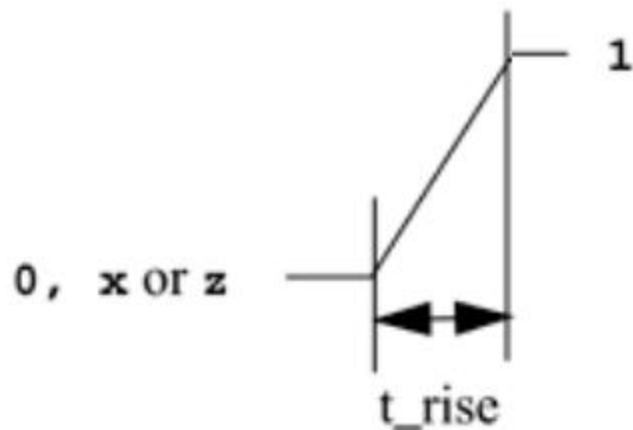
Gate-level Modeling - Example

```
module full_adder(sum, c_out, a, b, c_in);  
    output sum, c_out;  
    input a, b, c_in;  
    wire s1, c1, c2;  
  
    xor (s1, a, b);  
    and (c1, a, b);  
    xor (sum, s1, c_in);  
    and (c2, s1, c_in);  
    or (c_out, c2, c1);  
endmodule
```

Gate-level Modeling – Transition Delay

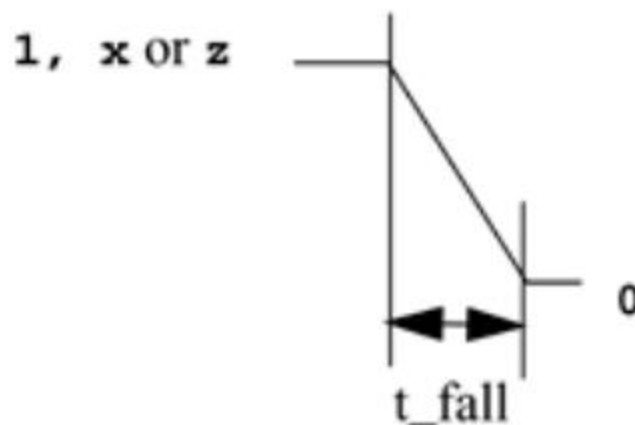
Rise Delay

- Transition to 1 from another value (0, x, z).



Fall Delay

- Transition to 0 from another value (1, x, z).



Turnoff Delay

- Transition to z from another value (0, 1, x).

Transition Delay Examples

buf b1 (out,in) - *rise = fall = turnoff = 0*

buf #(5) (out, in) - *rise = fall = turnoff = 5*

buf #(3, 5) (out, in) - *rise = 3, fall = 5, turnoff = 3*

buf #(3, 4, 5) (out, in) - *rise = 3, fall = 4, turnoff = 5*

Gate-level Modeling – Delay Settings

- Min/Typical/Max
- Example:
 - and #(2:3:4, 3:4:5, 4:5:6) a(out,in1,in2)

Dataflow Modeling

- Continuous Assignment
 - Driving a value onto a net
 - Example:
 - `assign <drive-strength>? <delay>? <list_of_assignments>;`
- Example:
 - `assign out = in1 & in2;`
 - `assign {c_out, sum[3:0]} = a[3:0] + b[3:0] + c_in;`
 - `assign #10 out = in1 & in2;`
- When does the simulator run the 'assign' statement?

Dataflow Modeling - Example

```
module adder_4_bit(sum, c_out, a, b, c_in);  
    output [3:0] sum;  
    output c_out;  
    input [3:0] a, b;  
    input c_in;  
  
    assign {c_out, sum} = a + b + c_in;  
endmodule
```

Behavioural Modeling - always vs. initial

initial

- Starts at $t=0$, runs once

initial

```
begin
  #1 $display ("Line 1");
  $display ("Line 2");
end
```

Sequential Block

always

- Starts at $t=0$, runs continuously

always

```
begin
  #1 $display ("Line 1");
  $display ("Line 2");
end
```

Behavioural Modeling - always vs. initial

initial

- Starts at $t=0$, runs once

initial

```
fork
  #1 $display ("Line 1");
  $display ("Line 2");
join
```

Parallel Block

always

- Starts at $t=0$, runs continuously

Behavioural Modeling - always vs. initial

initial

- Starts at $t=0$, runs once

```
initial
  fork
    #1 $display ("Line 1");
    $display ("Line 2");
  join
```

always

- Starts at $t=0$, runs continuously

```
initial
  clock = 1'b0;
always
  #10 clock = ~clock;
initial
  #1000 $finish;
```

- What happens to multiple initial and/or always blocks?

Behavioural Modeling – Assignment

- Blocking assignment (=)
 - Execute in the order they are specified
 - Block execution of statements that follow in a sequential block

```
initial begin
  i = 3;
  j = 4;
  #10 a = i + j;
  i = a + 5;
  j = a;
end
```

i = 12, j = 7

Behavioural Modeling – Assignment

- Blocking assignment (=)
 - Execute in the order they are specified
 - Block execution of statements that follow in a sequential block

```
initial begin
  i = 3;
  j = 4;
  #10 a = i + j;
  i = a + 5;
  j = a;
end
```

$i = 12, j = 7$

```
initial begin
  i = 3;
  j = 4;
  fork
    i = j;
    j = i;
  join
end
```

race condition

Behavioural Modeling – Assignment

- Blocking assignment (=)
 - Execute in the order they are specified
 - Block execution of statements that follow in a sequential block

```
initial begin
  i = 3;
  j = 4;
  #10 a = i + j;
  i = a + 5;
  j = a;
end
```

i = 12, j = 7

```
initial begin
  i = 3;
  j = 4;
  fork
    i = j;
    j = i;
  join
end
```

race condition

```
initial begin
  i = 3;
  j = 4;
  fork
    i = #1 j;
    j = #1 i;
  join
end
```

i = 4, j = 3

Behavioural Modeling – Assignment

```
initial begin
  i = 3;
  j = 4;
  begin
    i = #1 j;
    j = #1 i;
  end
end
```

i = 4, j = 4

Behavioural Modeling – Assignment

- Non-blocking assignments (\leq)
 - Allow scheduling of the following assignments

```
initial begin
  i = 3;
  j = 4;
  begin
    i = #1 j;
    j = #1 i;
  end
end
```

i = 4, j = 4

```
initial begin
  i = 3;
  j = 4;
  begin
    i <= #1 j;
    j <= #1 i;
  end
end
```

i = 4, j = 3

Behavioural Modeling – Timing Control

- Delay-based
 - Regular delay control
 - `#10 a = b & c;`
 - Intra-assignment delay control
 - `a = #10 b & c;`
 - Zero delay control
 - `#0 a = 0;`

Behavioural Modeling – Timing Control

- Event-based
 - @(clock)
 - @(posedge clock)
 - @(negedge clock)
 - @(sample_event)
 - Definition: event sample_event
 - Triggering: -> sample_event

Resources...

- [“Verilog HDL: A guide to digital design and synthesis” by Samir Palnitkar](#)
- Slides are posted in ~cs450
<http://www.student.cs.uwaterloo.ca/~cs450/w12/proj.shtml>