Bridge of Life Education

SOC Design
Verilog Delay Explained

Jiin Lai

Topics

1. Propagation Delay
2. Inertial / Transport Delay
3. Blocking / nonblocking
4. LHS (Inter) / RHS (Intra) delay
5. Procedure block / Continuous assignment

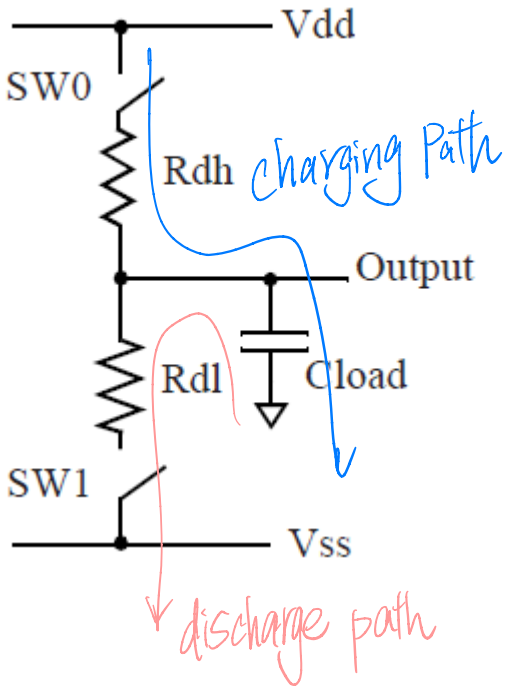
使用情况

Use the right delay model

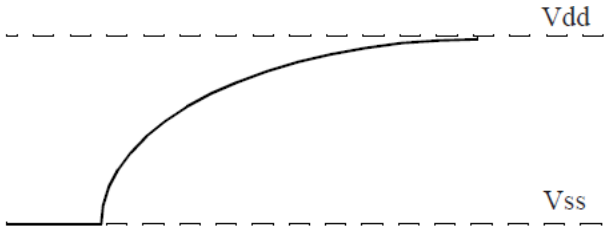
Switching Waveform

*(Rdh * Cload)* is called the **RC time constant**

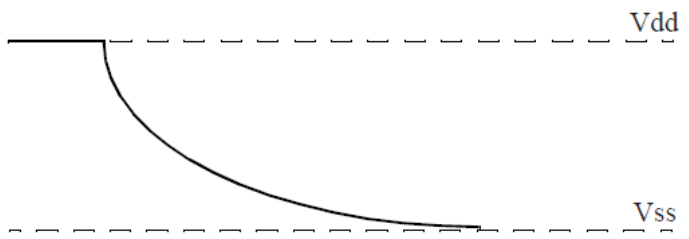
Inverter



$$V = Vdd * [1 - e^{-t/(Rdh * Cload)}]$$



$$V = Vdd * e^{-t/(Rdl * Cload)}$$

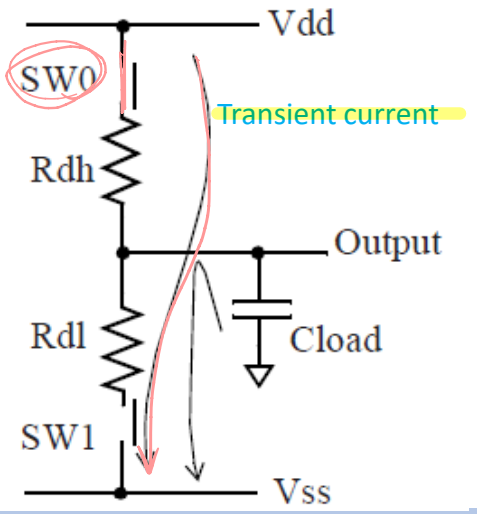


Switching from high to low

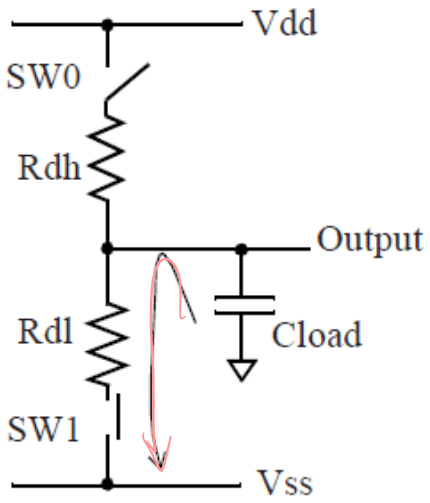
CMOS power { Switch Rate
leakage current << simulate current

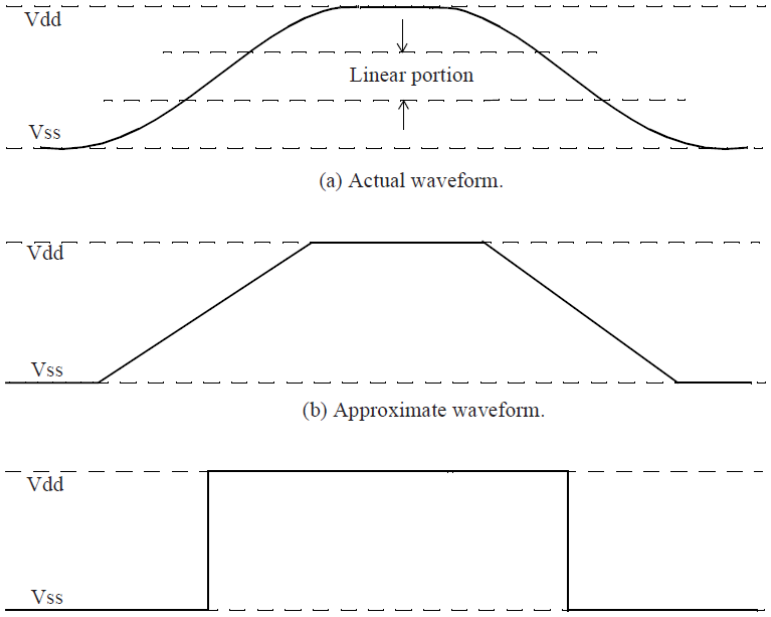
- When switching from high to low, **PMOS pull-up and the NMOS pull-down transistors are both on simultaneously for a short moment.**
- Later, the pull-up structure turns off the current flow from Cload to Vss.
- **After the output reaches the final state, there is no current flow** as the capacitance Cload is completely discharged.

Cell is switching
(pullup, pull-down both on)

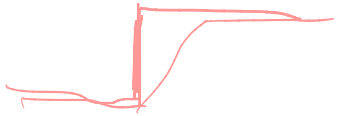


Discharging to logic-0
(pull-up off, pull-down on)

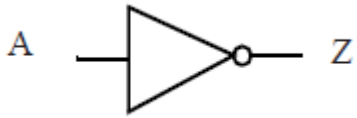


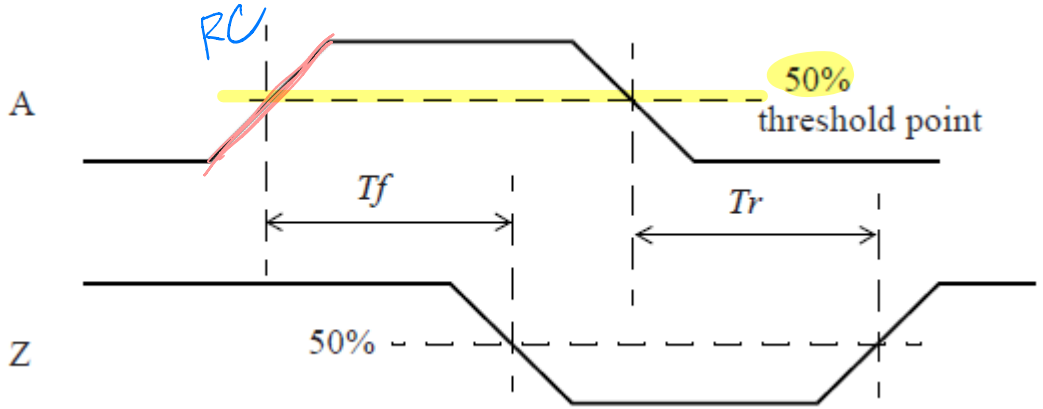


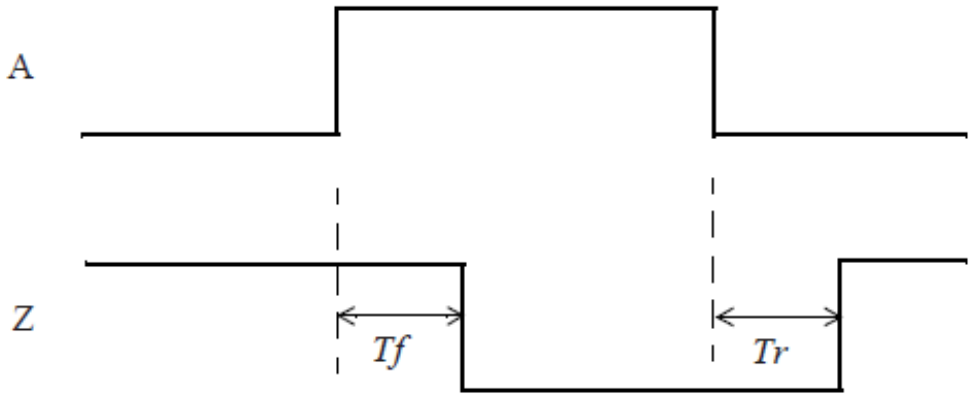
Propagation Delay



- Timing measurement on the switching waveforms.
  - Threshold point of an input falling edge: `**input_threshold_pct_fall** : 50.0;
  - Threshold point of an input rising edge: `**input_threshold_pct_rise** : 50.0;
  - Threshold point of an output falling edge: `**output_threshold_pct_fall** : 50.0;
  - Threshold point of an output rising edge: `**output_threshold_pct_rise** : 50.0;
- Propagation Delay:
  - Output fall delay (***Tf*** **)**
  - Output rise delay (***Tr***)







Inertial / Transport delay

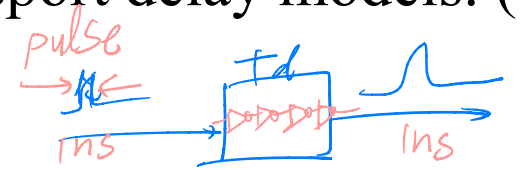
propergation → output.

- **Inertial delay** models only propagate signals to an output with pulse width greater than its propagation delay. If the pulse width is less, the early edge scheduled is replaced by the later edge. (Default)
  - **+pulse_r/100 +pulse_e/100**

    pw ≥ td unknown

pw < td ⇒ pw canceled

- **Transport delay** models propagate all signals to an output after any input signals change. Scheduled output value changes are queued for transport delay models. (Used for behavior model)
  - **+pulse_r/0 +pulse_e/0** (delay model)



- **Semi-realistic delay**
  - **+pulse_r/30 +pulse_e/70**

    pw ≤ 30% td → 看不到
    ≥ 70% td → 看得到
) 中間 → unknown

    reject pulses less than 30%,

    propagate unknowns for pulses between 30-70% and

    pass all pulses greater than 70% of propagation delay.

Inter (LHS) / Intra (RHS) Delay

left hand side

evaluate

Inter (LHS) Delay **#<delay> <LHS> = <RHS>**

The statement is executed after the delay. Most commonly used.

RHS ⇒ Transport delay

evaluate → delay → assign

Intra (RHS) Delay **<LHS> = #<delay> <RHS>**

The statement is evaluated with the signals on the RHS captured first. And the assignment to the LHS signal after the delay.

https://www.chipverify.com/verilog/verilog-inter-and-intra-assignment-delay#:~:text=An%20intra%2Dassignment%20delay%20is,only%20after%20the%20delay%20expires.

Category

8 組合

- Blocking / nonblocking
- LHS (Inter) / RHS (Intra) delay
- Procedure block / Continuous assignment
- Inertial / transport delay

	Blocking/RHS	Blocking/LHS	Nonblocking/RHS	Nonblocking/LHS
Procedure Block	X	**For Testbench**	**Transport Delay**	Not for Comb Logic
Continuous Assignment	Transport Delay ?	**Inertial Delay**	X	X

Procedure Block – Blocking - LHS (Inter) Delay

一般用bloking (for testbench，不用RTL)

Delay both the evaluation and update

Modeling Guideline: do not place delays on the LHS of blocking assignments to model combinational logic. This is a bad coding style.

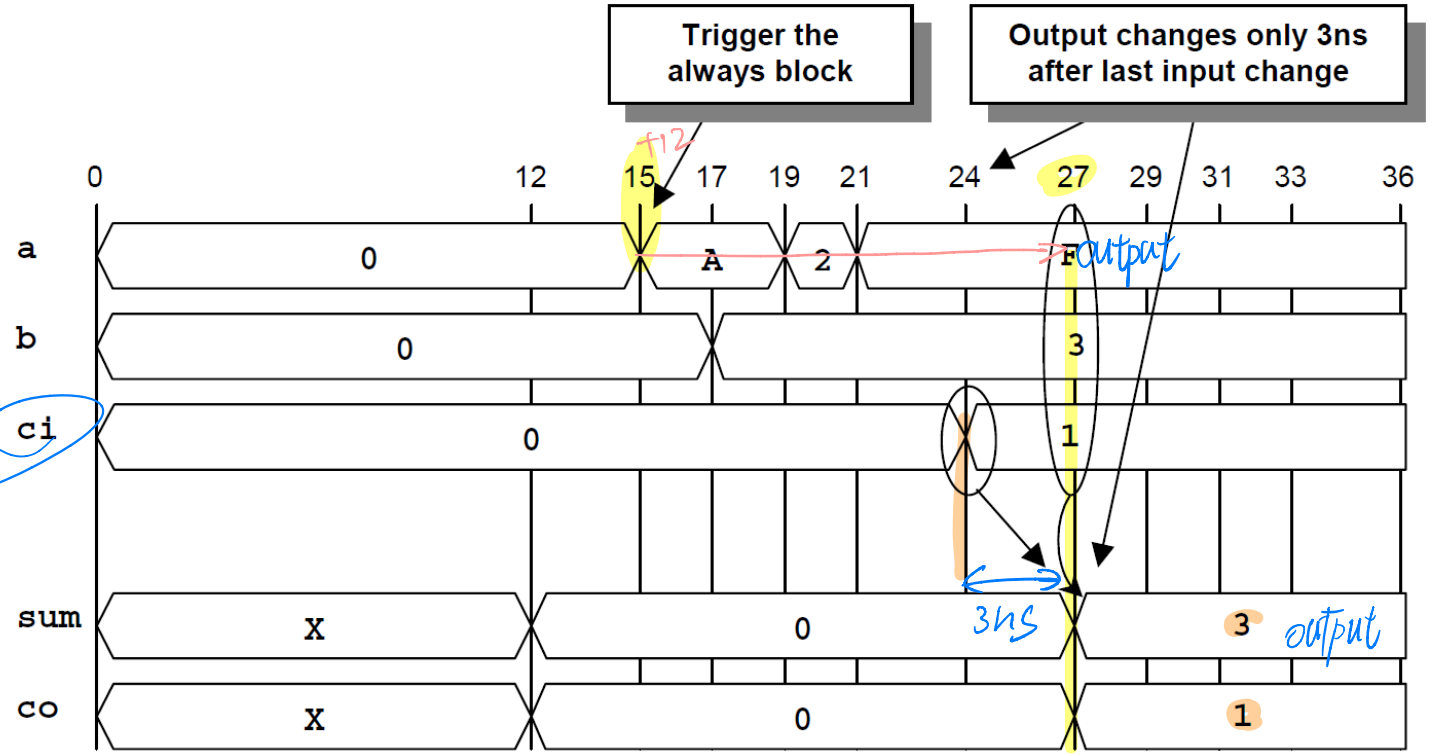
Testbench Guideline: placing delays on the LHS of blocking assignments in a testbench is reasonable since the delay is just being used to time-space sequential input stimulus events.

```
module adder_t1 (co, sum, a, b, ci);
  output        co;
  output  [3:0] sum;
  input   [3:0] a, b;
  input         ci;
  reg           co;
  reg     [3:0] sum;

  always @(a or b or ci)
    #12 {co, sum} = a + b + ci;
endmodule
```

```
always @(a or b or ci) begin
      tmp        = a + b + ci;
  #12 {co, sum} = tmp;
end
```

Used for testbench



Procedure Block – Blocking - RHS (Intra) Delay

不管

Delays the evaluation but not the update.
Delays the update but not the evaluation
Modeling Guideline: do not place delays on the RHS of blocking assignments to model combinational logic.
This is a bad coding style.
Testbench Guideline: do not place delays on the RHS of blocking assignments in a testbench.
General Guideline: placing a delay on the RHS of any blocking assignment is both confusing and a poor coding style. **This Verilog coding practice should be avoided.**

```
module adder_t6 (co, sum, a, b, ci);
  output       co;
  output [3:0] sum;
  input  [3:0] a, b;
  input        ci;
  reg          co;
  reg    [3:0] sum;

  always @(a or b or ci)
    {co, sum} = #12 a + b + ci;
endmodule
```

```
tmp        =      a + b + ci;
{co, sum}  = #12  tmp;
```

```
tmp        = #12  a + b + ci;
{co, sum}  =      tmp;
```

Do not use it

Procedure Block – Nonblocking - LHS (Inter) Delay

不用

Delay both the evaluation and the update.

Modeling Guideline: do not place delays on the LHS of nonblocking assignments to model combinational logic. This is a bad coding style

Testbench Guideline: nonblocking assignments are less efficient to simulate than blocking assignments;

```
module adder_t2 (co, sum, a, b, ci);
  output        co;
  output  [3:0] sum;
  input   [3:0] a, b;
  input         ci;
  reg           co;
  reg     [3:0] sum;

  always @(a or b or ci)
    #12 {co, sum} <=  a + b + ci;
endmodule
```

nonblk

Do not use it for Combinational Logic

Procedure Block / Nonblocking / RHS (Intra)

models combinational logic with transport delays. for delay-line logic

Delay the update but not the evaluation.

Modeling Guideline: place delays on the RHS of nonblocking assignments only when trying to **model transport output-propagation** behavior. This coding style will accurately model delay lines and combinational logic with pure transport delays; however, this coding style generally causes slower simulations. **Commonly use to model clock-to-output behavior on sequential logic.**

Testbench Guideline: This coding style is used in testbench when stimulus must be scheduled on future clock edges or after a set delay, while not blocking the assignment of subsequent stimulus events in the same procedural block.

```
module adder_t3 (co, sum, a, b, ci);
  output        co;
  output  [3:0] sum;
  input   [3:0] a, b;
  input         ci;
  reg           co;
  reg     [3:0] sum;

  always @(a or b or ci)
    {co, sum} <= #12 a + b + ci;
endmodule
```

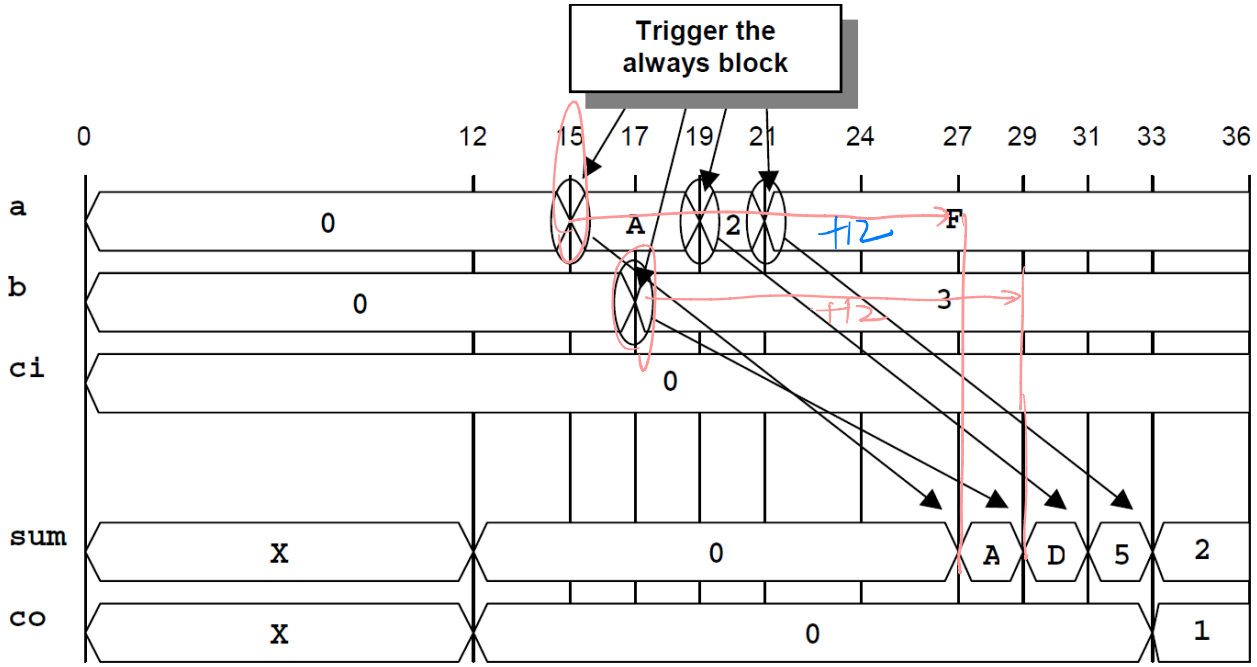
model Transport delays

non-blk /RHS

tmp needs to be in sensitivity list

```
always @(a or b or ci or tmp) begin
  tmp        <= #12 a + b + ci;
  {co, sum} <=      tmp;
end
```

```
always @(a or b or ci or tmp) begin
  tmp        <=      a + b + ci;
  {co, sum} <= #12 tmp;
end
```



Use for Transport Delay

Continuous assignment with delays

Only LHS, no non-blk

`assign y = ~a;`	Continuous assignment - no delay
`assign #5 y = ~a;`	Continuous assignment - LHS delay
~~`assign`~~ `y = #5 ~a;`	**Illegal** continuous assignment - RHS delay
~~`assign`~~ `y <= ~a;`	**Illegal** continuous nonblocking assignment

Continuous assignment – Blocking – LHS

Adding delays to continuous assignments accurately models combinational logic with inertial delays and is a recommended coding style.

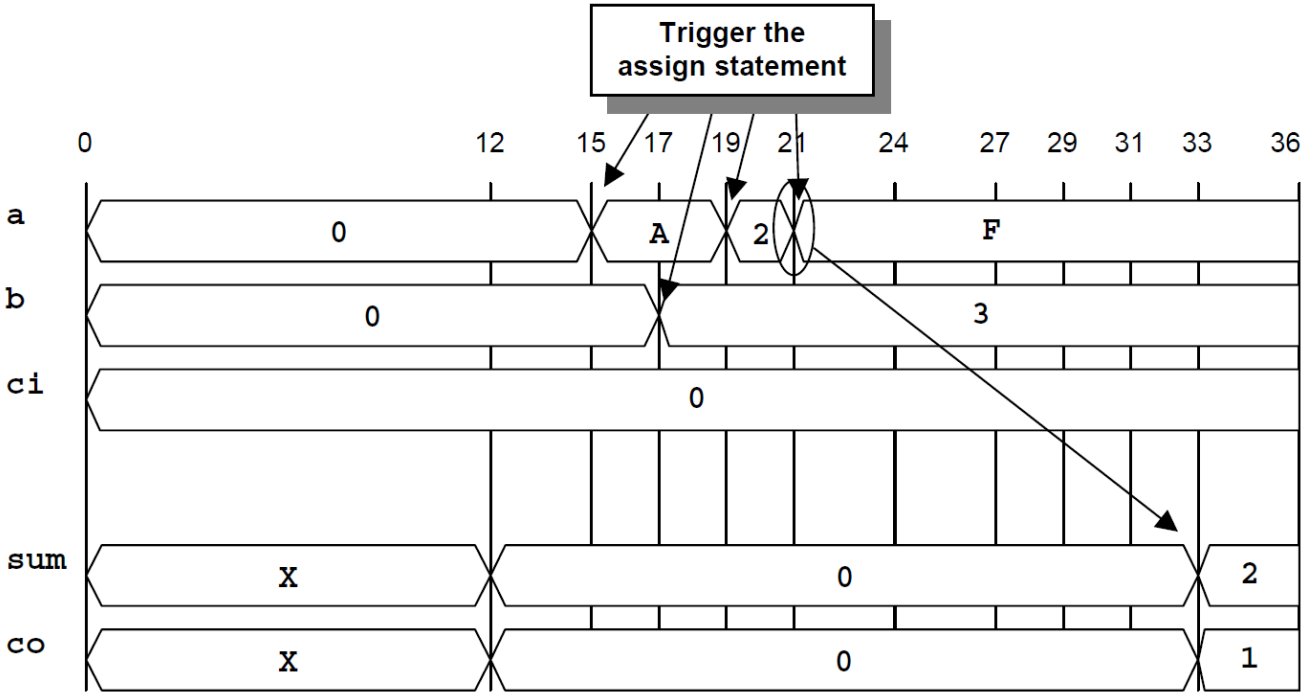
```
module adder_t4 (co, sum, a, b, ci);
  output        co;
  output [3:0]  sum;
  input  [3:0]  a, b;
  input         ci;

  assign #12 {co, sum} = a + b + ci;
endmodule
```

```
assign      tmp        = a + b + ci;
assign #12  {co, sum}  = tmp;
```

都可

```
assign #12  tmp        = a + b + ci;
assign      {co, sum}  = tmp;
```



Inertial Delay for Combinational Logic

Model Complex Combination logic with Mixed no-delay always blocks and delayed continuous assignments

Modeling Guideline: Use continuous assignments with delays to model simple combinational logic. This coding style will accurately model combinational logic with inertial delays.
Modeling Guideline: Use always blocks with no delays to model complex combinational logic using behavioral constructs suchas "case-casez-casex", "if-else", etc. Then, it is driven to output with continuous assignment with inertial delay.

```
module adder_t5 (co, sum, a, b, ci);
  output       co;
  output [3:0] sum;
  input  [3:0] a, b;
  input        ci;
  reg    [4:0] tmp;

  always @(a or b or ci) begin
    tmp = a + b + ci;
  end

  assign #12 {co, sum} = tmp;
endmodule
```

procedure block (no delay)

continuous assignment (+delay)