



EECS 2070 02 Fall 2021

# More on Coding Styles

黃稚存

Chih-Tsun Huang

[cthuang@cs.nthu.edu.tw](mailto:cthuang@cs.nthu.edu.tw)



國立清華大學  
資訊工程學系

Lecture 10

# 聲明

- ◎ 本課程之內容 (包括但不限於教材、影片、圖片、檔案資料等)，僅供修課學生個人合理使用，非經授課教師同意，不得以任何形式轉載、重製、散布、公開播送、出版或發行本影片內容 (例如將課程內容放置公開平台上，如 Facebook, Instagram, YouTube, Twitter, Google Drive, Dropbox 等等)。如有侵權行為，需自負法律責任。

# Outline

- ⦿ Initial Value to A Signal in RTL Code
- ⦿ Blocking and Non-Blocking Assignments
- ⦿ Combinational Loop
- ⦿ Sequential Blocks
- ⦿ Further Discussion

# Initial Value to A Signal in RTL Code

# Initial Value to A Variable

- For **wire-type** signals: equivalent to assign a constant

```
wire [4:0] address = 0;
```

- For **reg-type** signals: in general, **initial** block is forbidden (non-synthesizable) for RTL synthesis

```
reg data = 0;
```

Equivalent to

```
reg data;  
initial data = 0;
```

```
reg [1:0] state = 2'b00,  
          next_state = 2'b00;
```

- However, Vivado has specific support to these styles!!
  - Reset values to flip-flops
  - Initialize memory content (look-up table)

# DFF without Reset? (1/2)

- ⦿ Latest Vivado adjusted the initial value of reg-type variable to X
  - ◆ In its old version, the initial value is 0
- ⦿ There may be a mismatch between simulation and implementation
  - ◆ Especially if there is no reset input to a DFF
  - ◆ Simulation: the output of DFF is X
  - ◆ Implementation: the output of DFF is reset to 0 on the FPGA board

# DFF without Reset? (2/2)

- ◉ Vivado Design Suite User Guide: Synthesis (p.242)

- ◆ Latest Vivado accepts the following coding style

```
reg [3:0] count = 4'b0001;
```

- ◆ If count is a DFF's output, Vivado will synthesize the DFF with a global (board-level) reset implicitly

- ◉ So, you may use this style to design the clock divider for both the simulation and implementation

- ◆ Or you may add a reset input

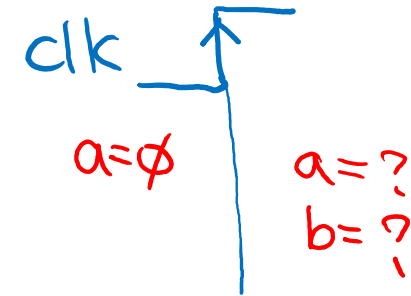
# Blocking and Non-Blocking Assignments



# Non-Blocking Assignments

- ◉ Simulator emulates parallel assignments with non-blocking statements

```
...  
  
always @(posedge clk)  
    a <= a + 1;  
  
always @(posedge clk)  
    b <= a;  
  
...
```

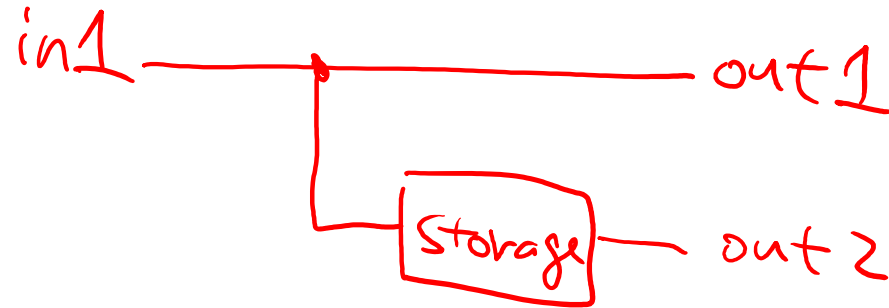


```
always @(posedge clk) begin  
    a <= a + 1;  
    b <= a;  
end
```

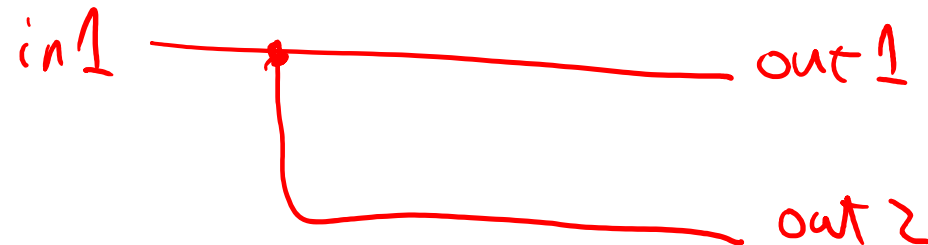
# Blocking Assignments

- Simulation result depends on statement order!

```
always @* begin
    out2 = out1;
    out1 = in1;
end
```



```
always @* begin
    out1 = in1;
    out2 = out1;
end
```

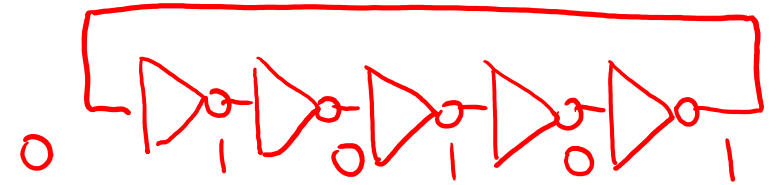


**\*Simulation may differ from synthesis result!!**

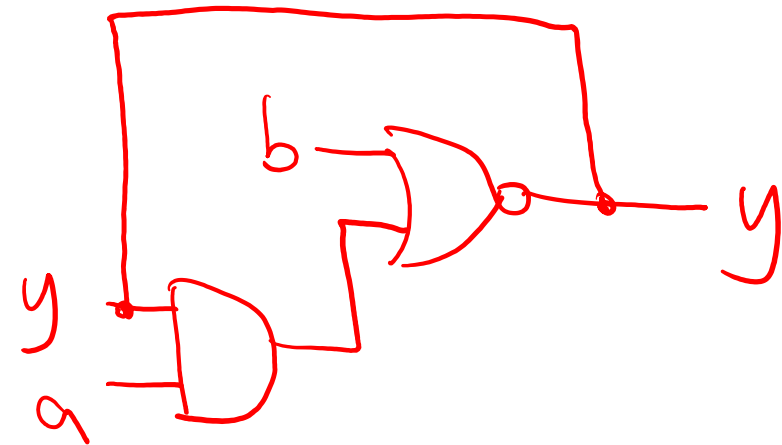
# Combinational Loop

# Combinational Loop

- In most cases, it is not what you want
  - ◆ Combinational circuit does not have feedback path, unlike sequential circuit
- May lead to oscillated output



```
...  
always @*  
    y = ~((y & a) | b);  
...
```



# More on Combinational Loop

- Combinaional loop: blocking vs. non-blocking?

```
always @(m, n)  
    m = m + n;
```

vs.

```
always @(m, n)  
    m <= m + n;
```

# Timing Arc in Synthesis

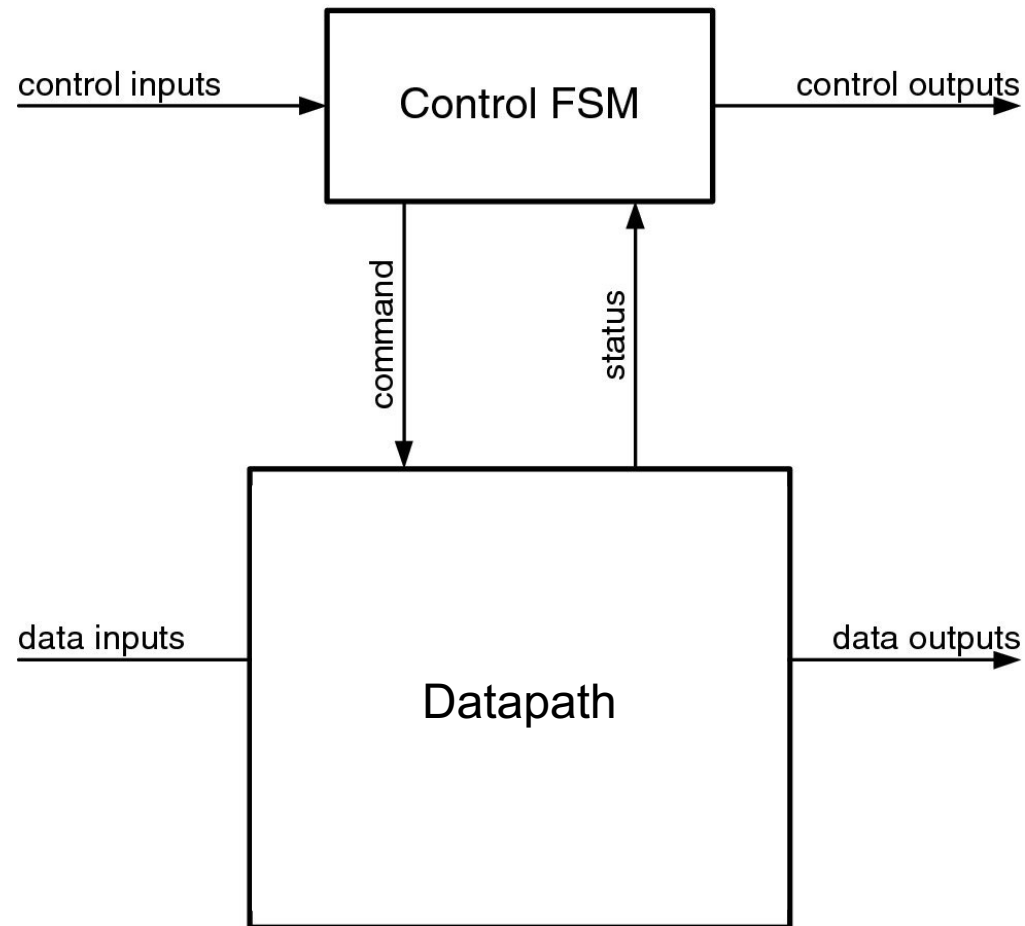
- ⦿ Timing arc is created when circular logic exists in your design
  - ◆ (Combinational) timing loop, or combinational loop
    - ▣ Information: Timing loop detected. (OPT-150)  
U3/A1 U3/Y U15/A U15/Y  
Warning: Disabling timing arc between pins 'A1' and 'Y' on cell 'U3' to break a timing loop. (OPT-314)
  - ◆ The synthesizer will break the loop itself
    - ▣ But this will likely prevent your synthesized code from operating correctly
- ⦿ Timing loop can be hard to find
  - ◆ It appears in gate-level description in synthesis
  - ◆ It may span across multiple always blocks (Design Compiler)  
`dc_shell> report_timing -loop`

# Vivado Synthesis Messages

- ⦿ [Synth 8-326] inferred exception to **break timing loop**:  
'set\_false\_path -through i\_4/i\_9/O'
- ⦿ [Synth 8-295] found **timing loop**.  
["C:/lab\_ex.sracs/sources\_1/imports/design and tb/lab\_ex.v":1]

# Timing Loop Can Be Hard to Find

- May span across multiple blocks



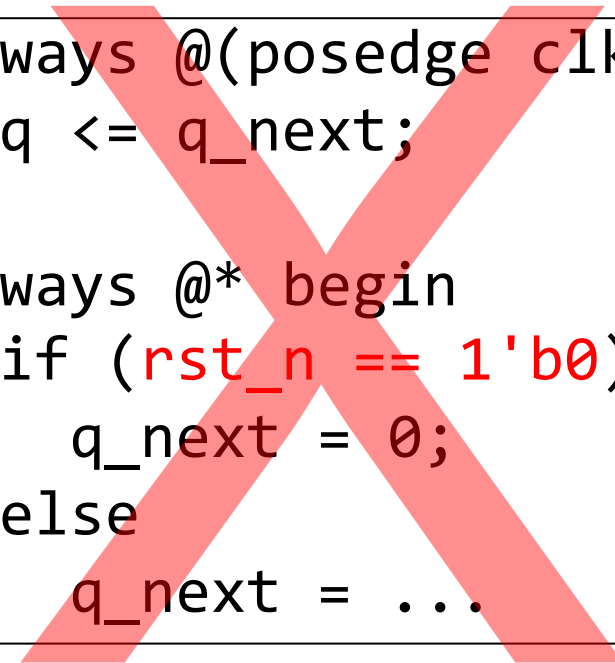


# Sequential Blocks

# DFF with Reset

- ⦿ D flip-flop (DFF) is also called **delay element**

```
...
reg q;
always @(posedge clk, negedge rst_n)
    if (rst_n == 1'b0)
        q <= 0;
    else
        q <= q_next;
```

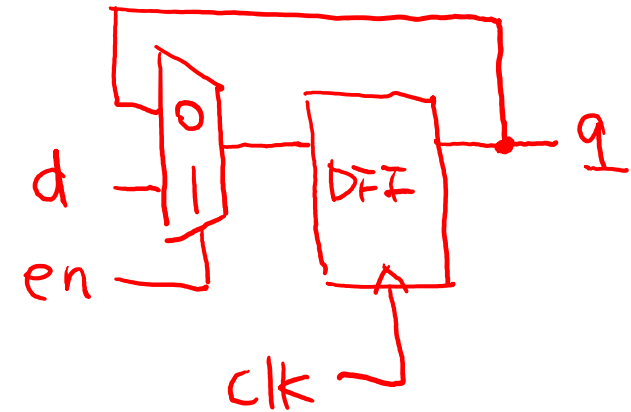


```
always @(posedge clk)
    q <= q_next;

always @* begin
    if (rst_n == 1'b0)
        q_next = 0;
    else
        q_next = ...
```

# Register with Enable (or Load) and Asynchronous Positive Reset

```
reg [7:0] q;  
always @(posedge clk, posedge reset) begin  
    if (reset == 1'b1) begin  
        q <= 0;  
    end else if (en == 1'b1) begin  
        q <= d;  
    end  
end
```



# Improper Style of Sequential **always**

```
reg q, r;
always @(posedge clk, negedge rst_n)
begin
    if (rst_n == 1'b0)
        q <= 0;
    else if (en == 1'b1)
        q <= q_next;
    if (rst_n == 1'b0)
        r <= 0;
    else if (en == 1'b1)
        r <= r_next;
end
```

```
reg q, r;
always @(posedge clk, negedge rst_n)
    if (rst_n == 1'b0) begin
        q <= 0;
        r <= 0;
    end else if (en == 1'b1) begin
        q <= q_next;
        r <= r_next;
    end
```

# Another Example

## Not Recommended

```
always @(posedge clk, posedge rst)
begin
    if (rst == 1'b1) begin
        R0 <= 0;
        R1 <= 0;
    end else begin
        if (L0 == 1'b1)
            R0 <= d0;
        if (L1 == 1'b1)
            R1 <= d1;
    end
end
```

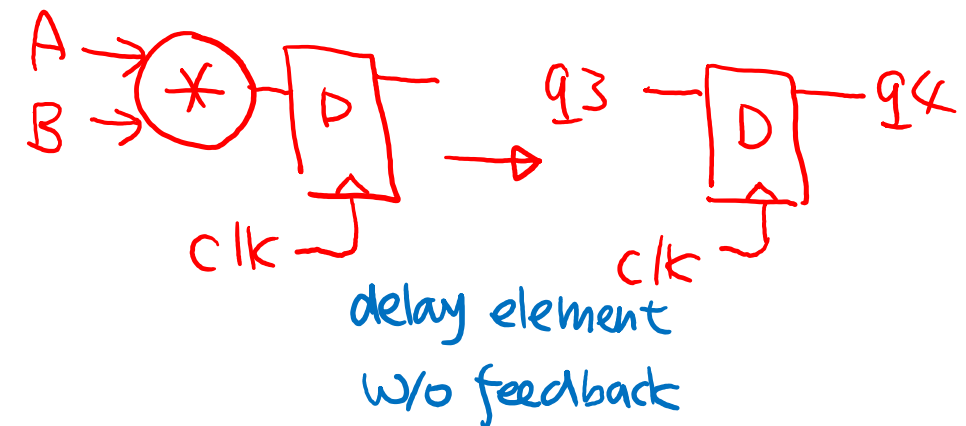
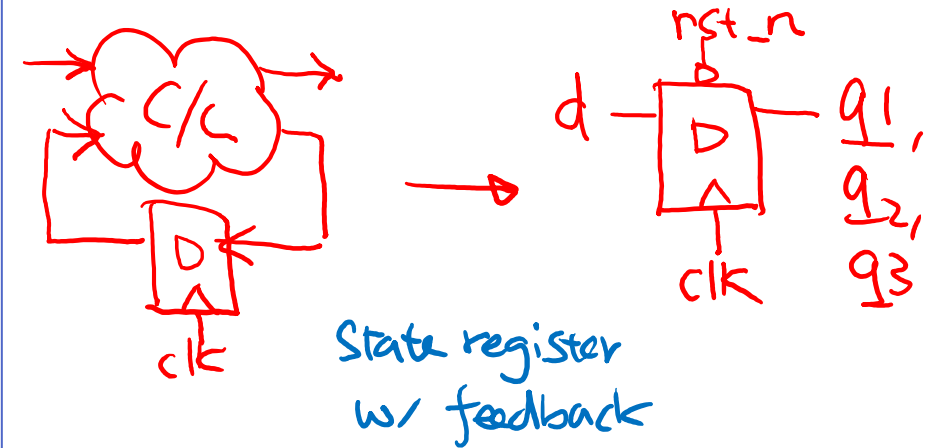
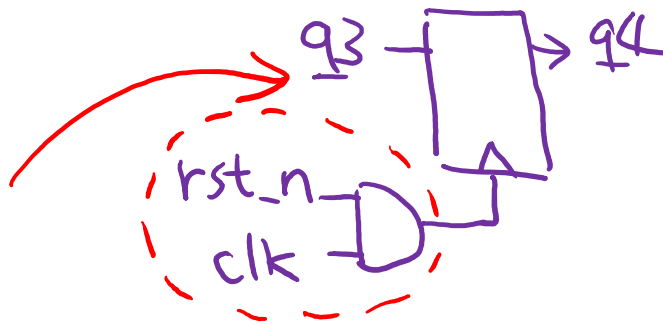
## Better

```
always @(posedge clk, posedge rst)
    if (rst == 1'b1)
        R0 <= 0;
    else if (L0 == 1'b1)
        R0 <= d0;

always @(posedge clk, posedge rst)
    if (rst == 1'b1)
        R1 <= 0;
    else if (L1 == 1'b1)
        R1 <= d1;
```

# Improper Mixture of FFs with and without Reset

```
always @ (posedge clk, negedge rst_n)
  if (rst_n == 1'b0) begin
    q1 <= 0;
    q2 <= 0;
    q3 <= 0;
  end else begin
    q1 <= ~q4;
    q2 <= q1;
    q3 <= q2;
    q4 <= q3;
  end
end
```



# Further Discussion

# Simulation or Lock Error in Vivado

- ⦿ ERROR: [Common 17-39] 'launch\_simulation' failed due to earlier errors
  - ◆ [USF-XSim-62] 'elaborate' step failed with error(s). Please check the Tcl console output or ['D:/lab/lab2/lab2.sim/sim\\_1/behav/xsim/elaborate.log'](#) file for more information.
  - ◆ [Vivado 12-4473] Detected error while running simulation. Please correct the issue and retry this operation.
- ⦿ The process cannot access the file because it is being used by another process:  
['D:/lab1/lab1.sim/sim\\_1/behav/xsim/elaborate.log'](#)