Hardware Modeling [VU] (191.011)
– WS25 –

Latches and combinational loops

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

Modified: 2025-12-16, 16:03 (f8a58e9)

Undesired Latches

- When should you use a latch or a flip-flop?

Undesired Latches

- When should you use a latch or a flip-flop?
- In this course latches are **always** undesired

Undesired Latches

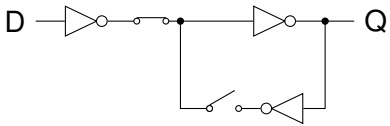
- When should you use a latch or a flip-flop?
- In this course latches are **always** undesired
- Often not available as primitive building blocks

Undesired Latches

- When should you use a latch or a flip-flop?
- In this course latches are **always** undesired
- Often not available as primitive building blocks
  - Built from combinational elements instead

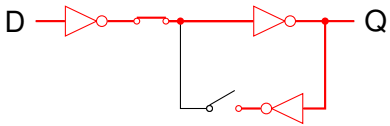
Undesired Latches

- When should you use a latch or a flip-flop?
- In this course latches are **always** undesired
- Often not available as primitive building blocks
  - Built from combinational elements instead



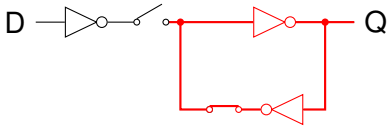
Undesired Latches

- When should you use a latch or a flip-flop?
- In this course latches are **always** undesired
- Often not available as primitive building blocks
  - Built from combinational elements instead



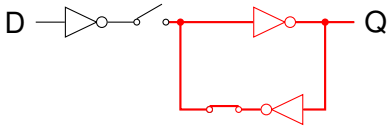
Undesired Latches

- When should you use a latch or a flip-flop?
- In this course latches are **always** undesired
- Often not available as primitive building blocks
  - Built from combinational elements instead



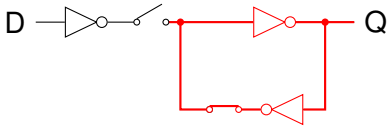
Undesired Latches

- When should you use a latch or a flip-flop?
- In this course latches are **always** undesired
- Often not available as primitive building blocks
  - Built from combinational elements instead
  - Delay of feedback path can become high



Undesired Latches

- When should you use a latch or a flip-flop?
- In this course latches are **always** undesired
- Often not available as primitive building blocks
  - Built from combinational elements instead
  - Delay of feedback path can become high
  - Feedback path can lead to oscillation



Pitfalls - Latch Errors (cont'd)

- How do latches *happen*?

Pitfalls - Latch Errors (cont'd)

- How do latches *happen*?

```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Latch Errors (cont'd)

- How do latches *happen*?

```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Latch Errors (cont'd)

- How do latches *happen*?

```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Latch Errors (cont'd)

- How do latches *happen*?
  - Not all paths write to signal

```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Latch Errors (cont'd)

- How do latches *happen*?
  - Not all paths write to signal
  - Holding state infer latch

```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

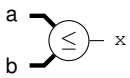
Pitfalls - Latch Errors (cont'd)

- How do latches *happen*?
  - Not all paths write to signal
  - Holding state ⇒ infer latch

```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Latch Errors (cont'd)

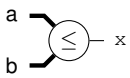
- How do latches *happen*?
  - Not all paths write to signal
  - Holding state ⇒ infer latch




```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Latch Errors (cont'd)

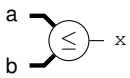
- How do latches *happen*?
  - Not all paths write to signal
  - Holding state ⇒ infer latch
- Always cover all paths/cases!




```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Latch Errors (cont'd)

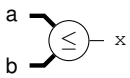
- How do latches *happen*?
  - Not all paths write to signal
  - Holding state ⇒ infer latch
- Always cover all paths/cases!
  - Default assignment or `others`




```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Latch Errors (cont'd)

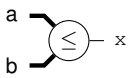
- How do latches *happen*?
  - Not all paths write to signal
  - Holding state ⇒ infer latch
- Always cover all paths/cases!
  - Default assignment or `others`




```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11     x <= '0';
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Latch Errors (cont'd)

- How do latches *happen*?
  - Not all paths write to signal
  - Holding state ⇒ infer latch
- Always cover all paths/cases!
  - Default assignment or `others`
- Detected during synthesis
  - **Never** ignore warnings!




```
1  entity cmp is
2    port(
3      a, b : in  unsigned;
4      x    : out std_ulogic
5    );
6  end entity;
7
8  architecture arch of cmp is
9  begin
10   main : process (all) begin
11     x <= '0';
12     if a <= b then
13       x <= '1';
14     end if;
15   end process;
16 end architecture;
```

Pitfalls - Combinational Loops

- Feedback loops not exclusive to latches

Pitfalls - Combinational Loops

- Feedback loops not exclusive to latches

```
1 architecture beh of counter is
2   signal cnt : unsigned(7 downto 0);
3 begin
4   comb : process(all) begin
5     cnt <= cnt + 1;
6   end process;
7 end architecture;
```

Pitfalls - Combinational Loops

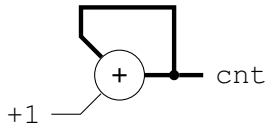
- Feedback loops not exclusive to latches

```
1 architecture beh of counter is
2   signal cnt : unsigned(7 downto 0);
3 begin
4   comb : process(all) begin
5     cnt <= cnt + 1;
6   end process;
7 end architecture;
```

Pitfalls - Combinational Loops

- Feedback loops not exclusive to latches

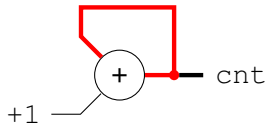
```
1 architecture beh of counter is
2   signal cnt : unsigned(7 downto 0);
3 begin
4   comb : process(all) begin
5     cnt <= cnt + 1;
6   end process;
7 end architecture;
```



Pitfalls - Combinational Loops

- Feedback loops not exclusive to latches

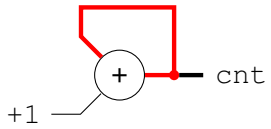
```
1 architecture beh of counter is
2   signal cnt : unsigned(7 downto 0);
3 begin
4   comb : process(all) begin
5     cnt <= cnt + 1;
6   end process;
7 end architecture;
```



Pitfalls - Combinational Loops

- Feedback loops not exclusive to latches
- Comb. process must never read **and** write to same signal

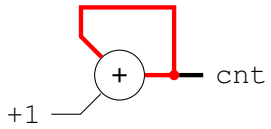
```
1 architecture beh of counter is
2   signal cnt : unsigned(7 downto 0);
3 begin
4   comb : process(all) begin
5     cnt <= cnt + 1;
6   end process;
7 end architecture;
```



Pitfalls - Combinational Loops

- Feedback loops not exclusive to latches
- Comb. process must never read **and** write to same signal
  - Hard to spot at interfaces

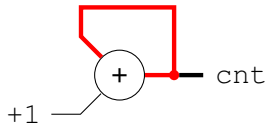
```
1 architecture beh of counter is
2   signal cnt : unsigned(7 downto 0);
3 begin
4   comb : process(all) begin
5     cnt <= cnt + 1;
6   end process;
7 end architecture;
```



Pitfalls - Combinational Loops

- Feedback loops not exclusive to latches
- Comb. process must never read **and** write to same signal
  - Hard to spot at interfaces
  - Feedback paths must contain flip-flops

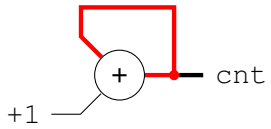
```
1 architecture beh of counter is
2   signal cnt : unsigned(7 downto 0);
3 begin
4   comb : process(all) begin
5     cnt <= cnt + 1;
6   end process;
7 end architecture;
```



Pitfalls - Combinational Loops

- Feedback loops not exclusive to latches
- Comb. process must never read **and** write to same signal
  - Hard to spot at interfaces
  - Feedback paths must contain flip-flops
- Also reported during synthesis

```
1 architecture beh of counter is
2   signal cnt : unsigned(7 downto 0);
3 begin
4   comb : process(all) begin
5     cnt <= cnt + 1;
6   end process;
7 end architecture;
```



Pitfalls - Combinational Loops (cnt'd)


```
1  architecture sync of counter is
2    signal cnt, cnt_next : unsigned(7 downto 0);
3  begin
4
5
6
7
8
9
10
11
12
13
14
15 end architecture;
```

Pitfalls - Combinational Loops (cnt'd)


```
1  architecture sync of counter is
2    signal cnt, cnt_next : unsigned(7 downto 0);
3  begin
4
5
6
7
8
9
10
11
12
13
14
15 end architecture;
```

Pitfalls - Combinational Loops (cnt'd)


```
1  architecture sync of counter is
2    signal cnt, cnt_next : unsigned(7 downto 0);
3  begin
4
5
6
7
8
9
10
11
12   comb : process(all) begin
13     cnt_next <= cnt + 1;
14   end process;
15 end architecture;
```

Pitfalls - Combinational Loops (cnt'd)


```
1  architecture sync of counter is
2    signal cnt, cnt_next : unsigned(7 downto 0);
3  begin
4    sync : process(clk, res_n) begin
5      if res_n = '0' then
6        cnt <= (others => '0');
7      elsif rising_edge(clk) then
8        cnt <= cnt_next;
9      end if;
10   end process;
11
12   comb : process(all) begin
13     cnt_next <= cnt + 1;
14   end process;
15 end architecture;
```

Pitfalls - Combinational Loops (cnt'd)

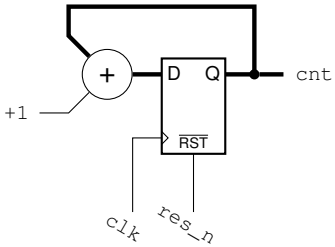

```
1  architecture sync of counter is
2    signal cnt, cnt_next : unsigned(7 downto 0);
3  begin
4    sync : process(clk, res_n) begin
5      if res_n = '0' then
6        cnt <= (others => '0');
7      elsif rising_edge(clk) then
8        cnt <= cnt_next;
9      end if;
10   end process;
11
12   comb : process(all) begin
13     cnt_next <= cnt + 1;
14   end process;
15 end architecture;
```

Pitfalls - Combinational Loops (cnt'd)


```
1  architecture sync of counter is
2    signal cnt, cnt_next : unsigned(7 downto 0);
3  begin
4    sync : process(clk, res_n) begin
5      if res_n = '0' then
6        cnt <= (others => '0');
7      elsif rising_edge(clk) then
8        cnt <= cnt_next;
9      end if;
10   end process;
11
12   comb : process(all) begin
13     cnt_next <= cnt + 1;
14   end process;
15 end architecture;
```

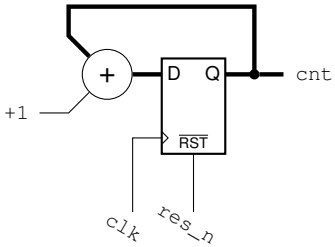
Pitfalls - Combinational Loops (cnt'd)


```
1  architecture sync of counter is
2    signal cnt, cnt_next : unsigned(7 downto 0);
3  begin
4    sync : process(clk, res_n) begin
5      if res_n = '0' then
6        cnt <= (others => '0');
7      elsif rising_edge(clk) then
8        cnt <= cnt_next;
9      end if;
10   end process;
11
12   comb : process(all) begin
13     cnt_next <= cnt + 1;
14   end process;
15 end architecture;
```



Pitfalls - Combinational Loops (cnt'd)


```
1  architecture sync of counter is
2    signal cnt, cnt_next : unsigned(7 downto 0);
3  begin
4    sync : process(clk, res_n) begin
5      if res_n = '0' then
6        cnt <= (others => '0');
7      elsif rising_edge(clk) then
8        cnt <= cnt_next;
9      end if;
10   end process;
11
12   comb : process(all) begin
13     cnt_next <= cnt + 1;
14   end process;
15 end architecture;
```



Summary

- Latches
  - Often due to missing default values

Summary

- Latches
  - Often due to missing default values
- Comb. loops
  - Never read **and** write same signal

Summary

- Latches
  - Often due to missing default values
- Comb. loops
  - Never read **and** write same signal
- Never ignore tool warnings!

```
10631 VHDL Process Statement warning at top_arch.vhd(13): inferring latch(es) for signal or variable "abc", which holds its previous value in one or more paths through the process

332125 Found combinational loop of 2 nodes
```

Lecture Complete!