香港中文大學
The Chinese University of Hong Kong

CENG3430 Rapid Prototyping of Digital Systems

Appendix: Use of Signal and Variable

Ming-Chang YANG

mcyang@cse.cuhk.edu.hk

Outline

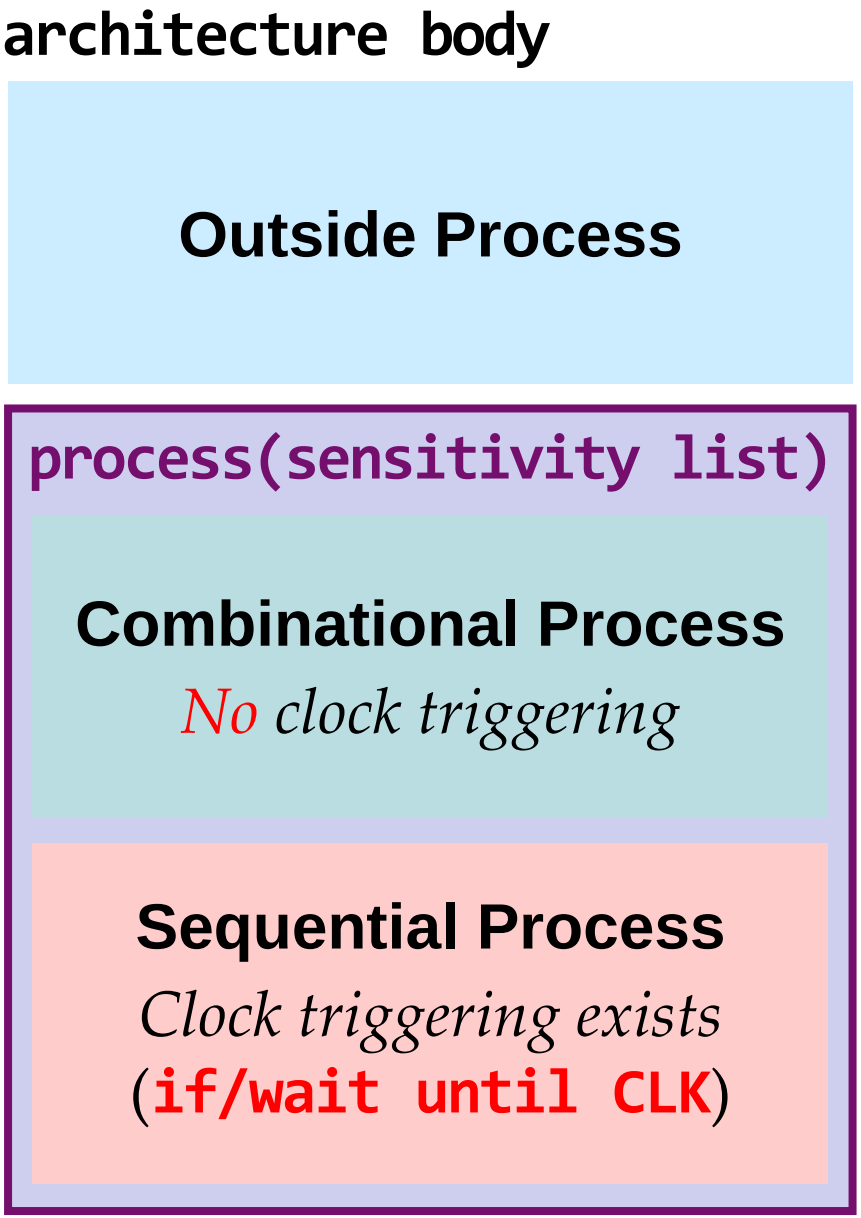
- Use of Signal and Variable
  - **Outside Process**:

    Concurrent Statements

  - **Inside Process**:

    Sequential Statements

    - **Combinational Process**
      - Has **no `CLK`** triggering
    - **Sequential Process**
      - Has **`CLK`** triggering



Outside Process

- **Signal assignments outside a process**
  - All the statements outside processes are concurrent and will be executed once whenever any RHS signal changes.

```
architecture test_arch of test is
  out1 <= in1 and in2; -- concurrent statement
  out2 <= in1 or in2;  -- concurrent statement
end test_arch;
```

- **Variable assignments outside a process**
  - Variables can only live *inside* processes!

- **Signal assignments outside a process:**
  All the statements outside processes will be executed once whenever any RHS signal changes.
- **Variable assignments outside a process :**
  Variables can only live *inside* processes!

Combinational Process

- A **combinational process** will be executed once whenever any signal in the sensitivity list changes.
  - No clock triggering can be found inside.

```
process(in1, in2) -- sensitivity list
  variable v1, v2: std_logic;
  begin
    s1 <= in1 and in2;
    s1 <= in1 or in2;  -- the last asgmt. for s1
    v1 := in1 and in2;
    v1 := in1 or in2;
end process
```

- **Signal assignments inside a combinational process:**
  Only ***the last*** assignment for a particular signal takes effect.
- **Variable assignments inside a combinational process:**
  All assignments take effect immediately and sequentially.

Class Exercise 1

Student ID: _____ Date:
Name: _____ _____

```
 1 signal S1, S2: bit;
 2 signal S_OUT: bit_vector(1 to 8);
 3 process (S1, S2)
 4 variable V1, V2: bit;
 5 begin
 6   V1 := '1';
 7   V2 := '1';
 8   S1 <= '1';
 9   S2 <= '1';
10   S_OUT(1) <= V1;
11   S_OUT(2) <= V2;
12   S_OUT(3) <= S1;
13   S_OUT(4) <= S2;
14   V1 := '0';
15   V2 := '0';
16   S2 <= '0';
17   S_OUT(5) <= V1;
18   S_OUT(6) <= V2;
19   S_OUT(7) <= S1;
20   S_OUT(8) <= S2;
21 end process;
```

- Which line(s) will NOT take effect?
  Answer:

  _____

- When will the process be executed?
  Answer:

  _____

- What are the values of S_OUT after execution?

S_OUT(1): S_OUT(5):
S_OUT(2): S_OUT(6):
S_OUT(3): S_OUT(7):
S_OUT(4): S_OUT(8):

Class Exercise 1 (Answer)

```
 1 signal S1, S2: bit;
 2 signal S_OUT: bit_vector(1 to 8);
 3 process (S1, S2)
 4 variable V1, V2: bit;
 5 begin
 6   V1 := '1';
 7   V2 := '1';
 8   S1 <= '1';
 9   S2 <= '1'; -- Has no effect
10   S_OUT(1) <= V1; -- Assigns '1'
11   S_OUT(2) <= V2; -- Assigns '1'
12   S_OUT(3) <= S1; -- Assigns '1'
13   S_OUT(4) <= S2; -- Assigns '0'
14   V1 := '0';
15   V2 := '0';
16   S2 <= '0'; -- Overrides Line 9
17   S_OUT(5) <= V1; -- Assigns '0'
18   S_OUT(6) <= V2; -- Assigns '0'
19   S_OUT(7) <= S1; -- Assigns '1'
20   S_OUT(8) <= S2; -- Assigns '0'
21 end process;
```

- Which line(s) will NOT take effect?
  Answer:
  Line 9
- When will the process be executed?
  Answer:
  When S1 or S2 changes
- What are the values of S_OUT after execution?

S_OUT(1):'1' S_OUT(5):'0'
S_OUT(2):'1' S_OUT(6):'0'
S_OUT(3):'1' S_OUT(7):'1'
S_OUT(4):'0' S_OUT(8):'0'

Sequential/Clocked Process

- A **sequential process** will be executed on clock edges regularly or when async. condition satisfies.
  - A clock edge detection can be found inside.

```
process(sensitivity list)
begin
   ... -- same as a combinational process
   if ( risign_edge(clk) ) then
     out1 <= in1 and in2; -- "<=" works like a flip-flop
   end if;
   ... -- same as a combinational process
end process;
```

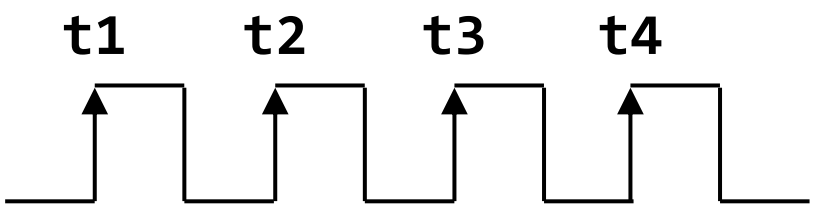
- **Signal assignments inside a sequential process:**
  Only ***the last*** assignment for a particular signal takes effect;
  <= is a flip-flop: The assignment takes effect on the next edge.
- **Variable assignments inside a sequential process :**
  All assignments take effect immediately and sequentially.

Class Exercise 2

Student ID: _____ Date:
Name: _____ _____

- Find the signal results after clock edges **t1 ~ t4**:

```
signal s1: integer:=1;
signal s2: integer:=2;
signal s3: integer:=3;
...
process
begin
wait until rising_edge(clk);
  s1 <= s2 + s3;
  s2 <= s1;
  s3 <= s2;
  sum <= s1 + s2 + s3;
end process
end
```

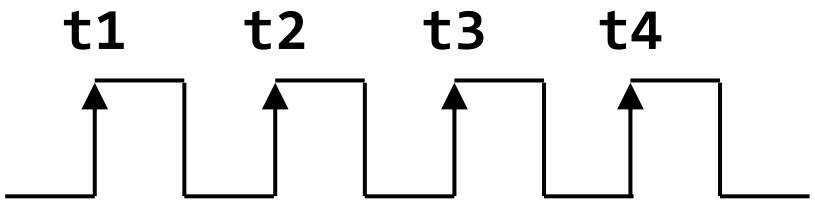


	t1	t2	t3	t4
s1				
s2				
s3				
sum				

Class Exercise 2 (Answer)

- Find the signal results after clock edges **t1 ~ t4**:

```
signal s1: integer:=1;
signal s2: integer:=2;
signal s3: integer:=3;
...
process
begin
wait until rising_edge(clk);
   s1 <= s2 + s3;
   s2 <= s1;
   s3 <= s2;
   sum <= s1 + s2 + s3;
end process
```



	t1	t2	t3	t4
s1	2+3	1+2	5+1	3+5
s2	1	5	3	6
s3	2	1	5	3
sum	1+2 +3	5+1 +2	3+5 +1	6+3 +5

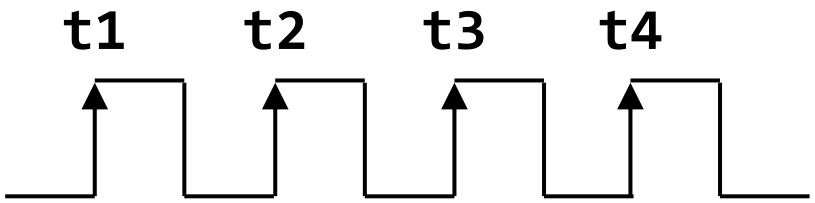
- **Signal assignments inside a sequential process:**
  <= is a flip-flop: The assignment takes effect on the next edge.

Class Exercise 3

Student ID: _____ Date:
Name: _____ _____

- Find the signal results after clock edges **t1 ~ t4**:

```
process
variable v1: integer:=1;
variable v2: integer:=2;
variable v3: integer:=3;
begin
wait until rising_edge(clk);
  v1 := v2 + v3;
  v2 := v1;
  v3 := v2;
  sum <= v1 + v2 + v3;
end process
end
```

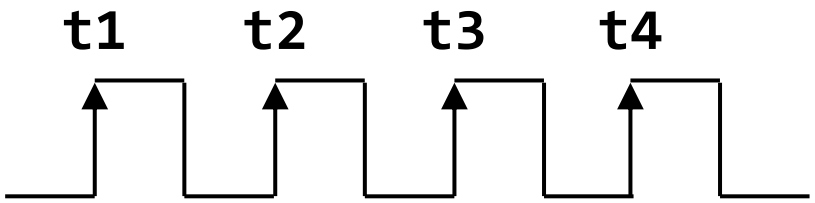


	t1	t2	t3	t4
v1				
v2				
v3				
sum				

Class Exercise 3 (Answer)

- Find the signal results after clock edges **t1 ~ t4**:

```
process
variable v1: integer:=1;
variable v2: integer:=2;
variable v3: integer:=3;
begin
wait until rising_edge(clk);
   v1 := v2 + v3;
   v2 := v1;
   v3 := v2;
   sum <= v1 + v2 + v3;
end process
end
```



	t1	t2	t3	t4
v1	2+3	5+5	10+10	20+20
v2	5	10	20	40
v3	5	10	20	40
sum	15	30	60	120

➢ **Variable assignments inside a sequential process :**

All assignments take effect immediately and sequentially.

Summary

- **Signal assignments outside a process:**
  All the statements outside processes will be executed once whenever any RHS signal changes.
- **Variable assignments outside a process :**
  Variables can only live ***inside*** processes!

- **Signal assignments inside a combinational process:**
  Only ***the last*** assignment for a particular signal takes effect.
- **Variable assignments inside a combinational process:**
  All assignments take effect immediately and sequentially.

- **Signal assignments inside a sequential process:**
  Only ***the last*** assignment for a particular signal takes effect;
  <= is a flip-flop: The assignment takes effect on the next edge.
- **Variable assignments inside a sequential process :**
  All assignments take effect immediately and sequentially.