Signals & Variables (1A)

Concurrent & Sequential Signal Assignments

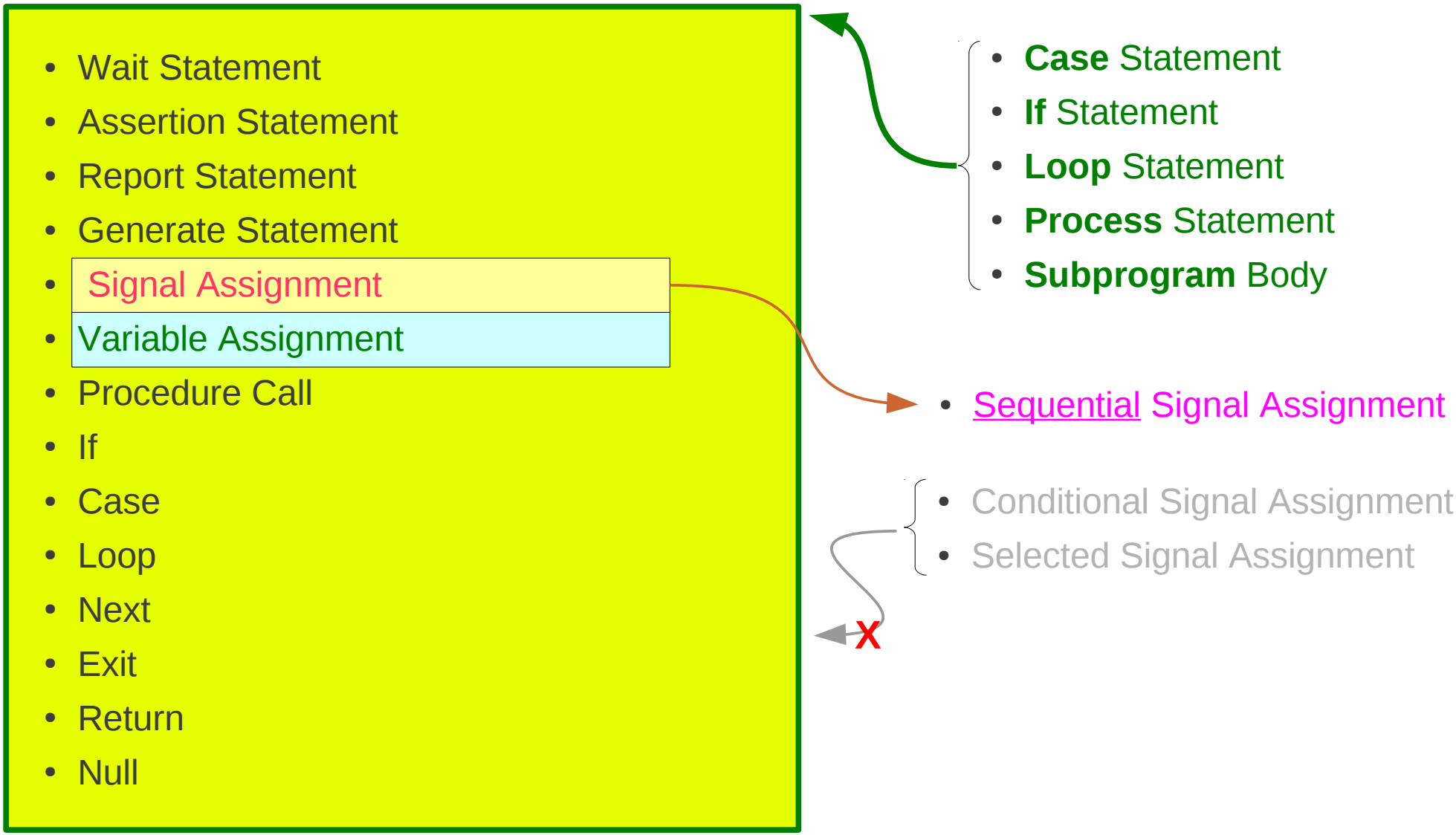
Young Won Lim
05/23/2012

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Young Won Lim
05/23/2012

Sequential Statement



Concurrent Statement

- Block Statement
- Process Statement
- Component Statement
- Generate Statement
- **Concurrent** Signal Assignment
- Concurrent Assertion
- Concurrent Procedure Call

- **Architecture** Body
- **Block** Statement
- **Generate** Statement

- **Conditional** Signal Assignment
- **Selected** Signal Assignemnt

Concurrent Signal Assignment

- **Conditional** Signal Assignment

```
Z <=    A or B    [ after 1 ns ]   when SEL = "00"  else
        A or C    [ after 2 ns ]   when SEL = "01" else
        A or D    [ after 2 ns ]   when SEL = "10" else
        A or E    [ after 3 ns ]   when SEL = "11" else
        A or F    [ after 4 ns ]   ;
```

condition

- **Selected** Signal Assignment

```
with SEL select
Z <=    A or B    [ after 1 ns ]   when "00" ,
        A or C    [ after 2 ns ]   when "01" ,
        A or D    [ after 3 ns ]   when "10" ,
        A or E    [ after 4 ns ]   when "11" ,
        A or F    [ after 5 ns ]   when others;
```

selection

Conditional Signal Assignment (1)

```
Z <=   A or B   [ after 1 ns ]  ;
```

simple concurrent statement

```
Z <=   A or B   [ after 1 ns ]  when S0 = '1' ;
```

One condition

```
Z <=   A or B   [ after 1 ns ]  when S0 = '1' else
       C or D   [ after 2 ns ]  ;
```

One condition with 'else'

```
Z <=   A or B   [ after 1 ns ]  when S0 = '1'  else
       C or D   [ after 2 ns ]  when S1 = '1' else
       E or F   [ after 3 ns ]  ;
```

Two conditions with 'else'

Concurrent Signal Assignment

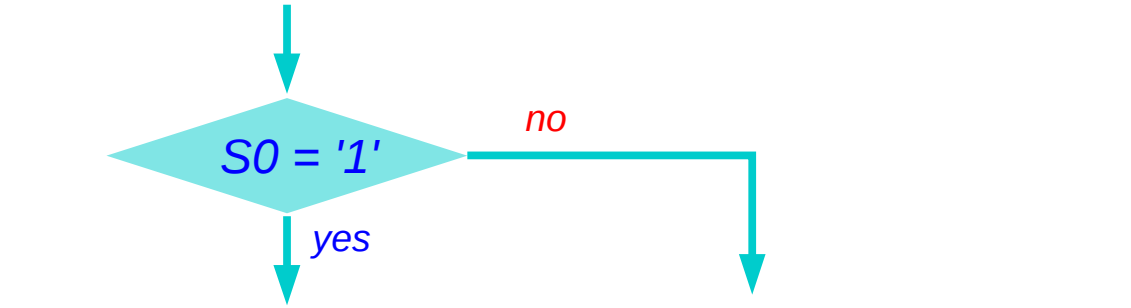
- **Conditional** Signal Assignment
- **Selected** Signal Assignment

Conditional Signal Assignment (2)

```
Z <=   A or B   [ after 1 ns ]   ;
```
simple concurrent statement

```
Z <=   A or B   [ after 1 ns ]   when S0 = '1' ;
```
One condition

```
Z <=   A or B   [ after 1 ns ]   when S0 = '1' else
       C or D   [ after 2 ns ]   ;
```
One condition with 'else'



Conditional Signal Assignment (3)

```
Z <=   A or B   [ after 1 ns ]   ;
```

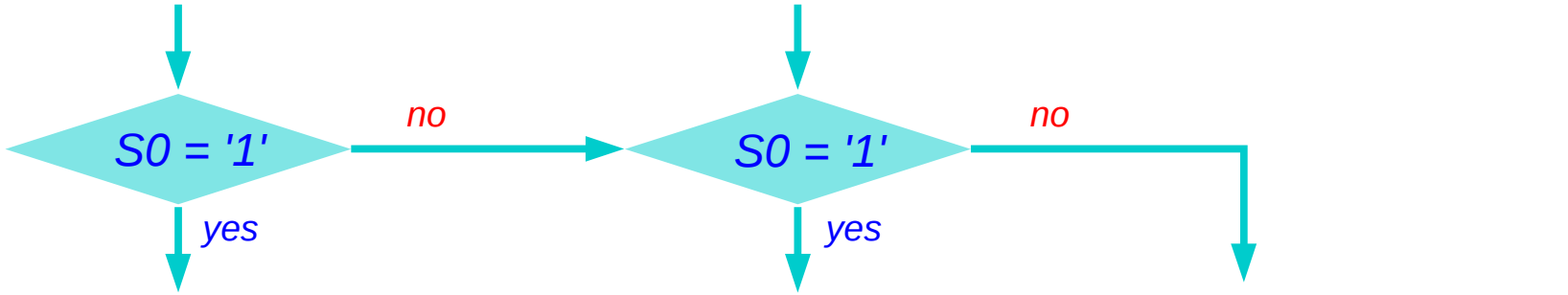
simple concurrent statement

```
Z <=   A or B   [ after 1 ns ]   when S0 = '1' ;
```

One condition

```
Z <=   A or B   [ after 1 ns ]   when S0 = '1'  else
       C or D   [ after 2 ns ]   when S1 = '1' else
       E or F   [ after 3 ns ]   ;
```

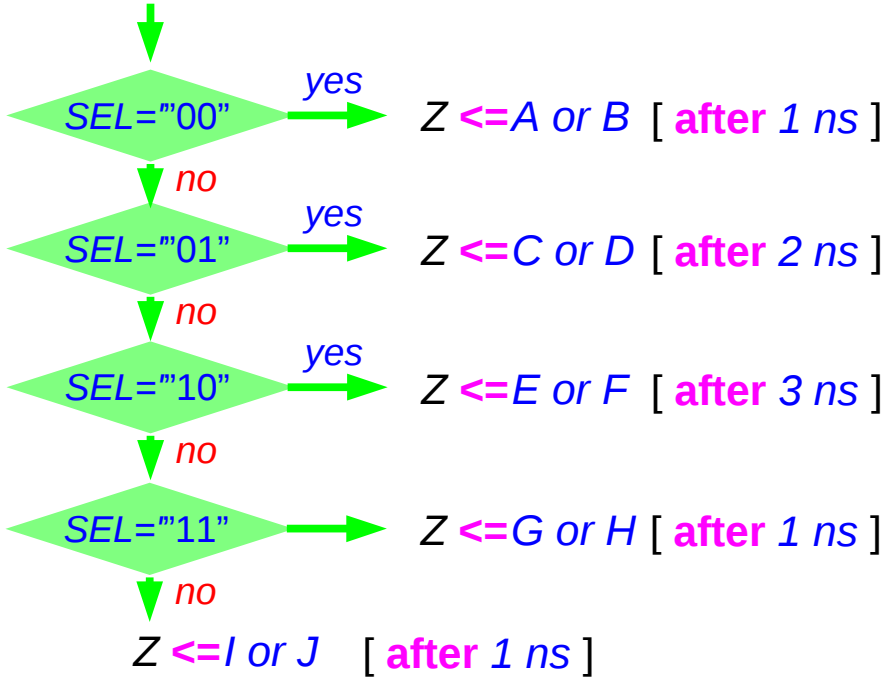
Two conditions with 'else'



Selected Signal Assignment

```
with SEL select
Z <=    A or B   [ after 1 ns ]  when "00" ,
        C or D   [ after 2 ns ]  when "01" ,
        E or F   [ after 3 ns ]  when "10" ,
        G or H   [ after 4 ns ]  when "11" ,
        I or J   [ after 5 ns ]  when others;
```

selection



Simulation Time (1)

Unit: ms, ns, ps, …

$1\,ms = 1000\,ns$

$1\,ns = 1000\,ps$

Real Delay

Unitless Delta D

used for a simulator to mimic parallel activities simulator

$1\,ps \neq n \cdot \Delta$

no integer n that make n delta's equal to 1 ps.

$n \cdot \Delta = 0\,ps = 0\,ns \cdots$

Zero Delay

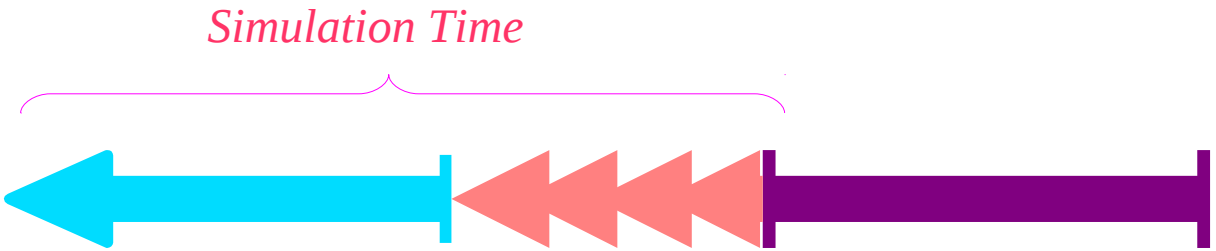
Assumed to take no time

Not related to any time

Zero Delay Assignment

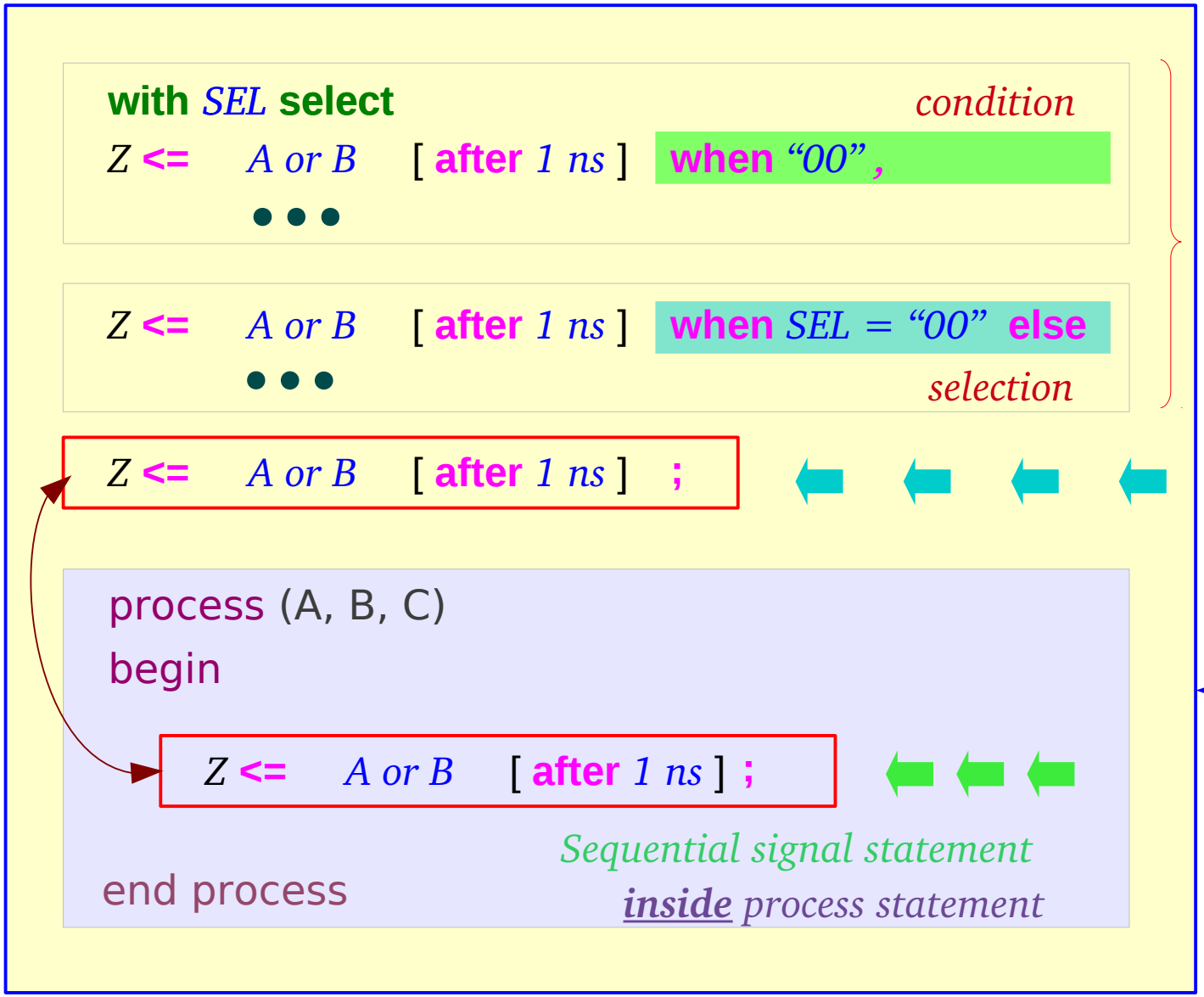
```
X1 <= A or B ;
X1 <= A or B after 0 ns;
```

Simulation Time (2)

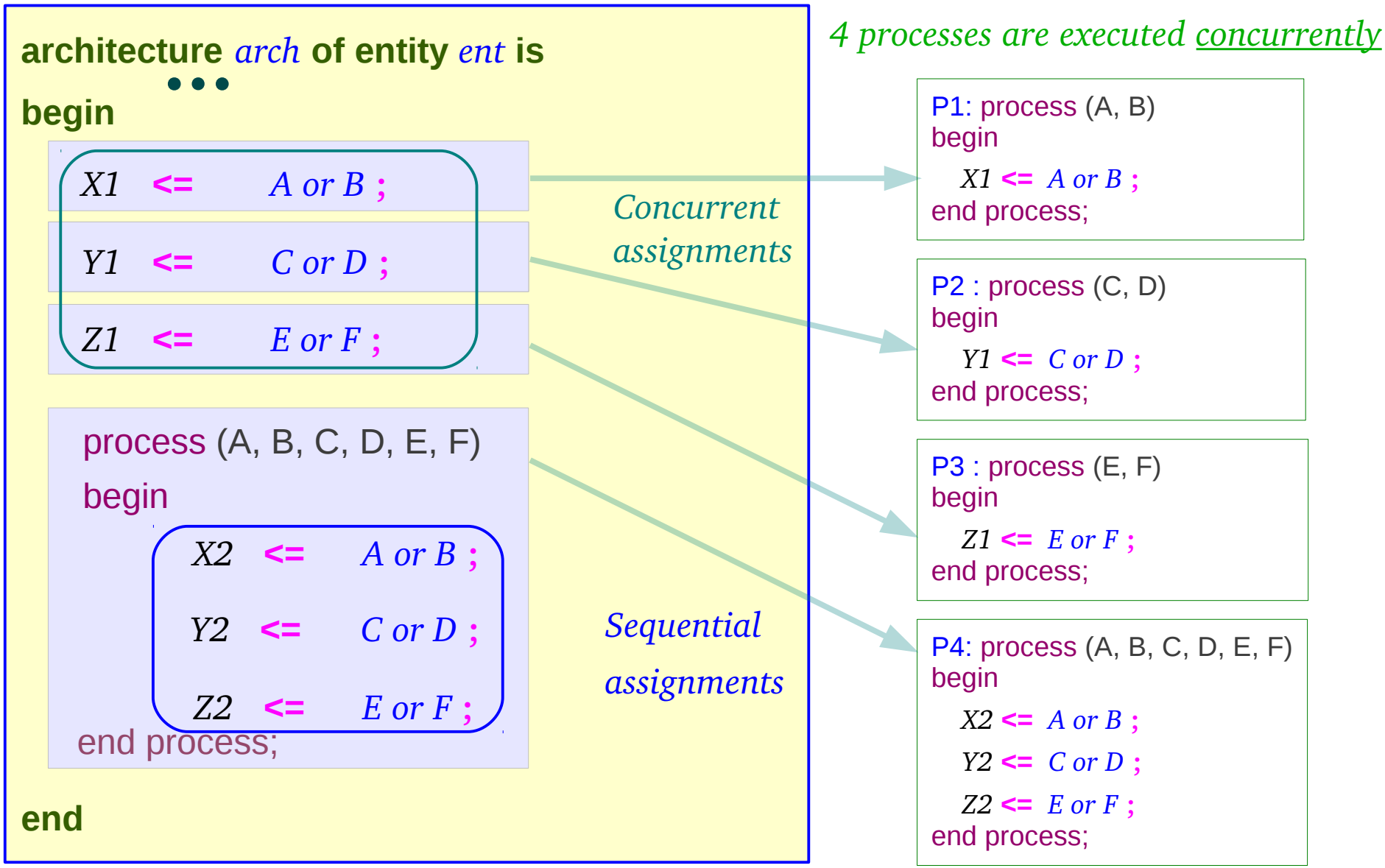


Unit: ms, ns, ps, …	Unitless Delta	
		Zero Delay Assignment
X1	<=	A or B ;
X1 is updated after at least one D		**Non-zero Delay Assignment**
X1	<=	A or B **after** 2 ns;
X1 is updated after 2 ns		

Concurrent vs Sequential (1)



Concurrent vs Sequential (2)



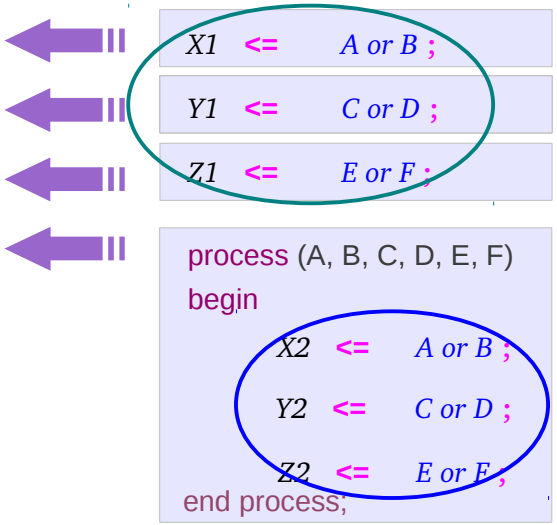
Concurrent vs Sequential (3)

Simulation of parallel activities

4 processes are executed concurrently

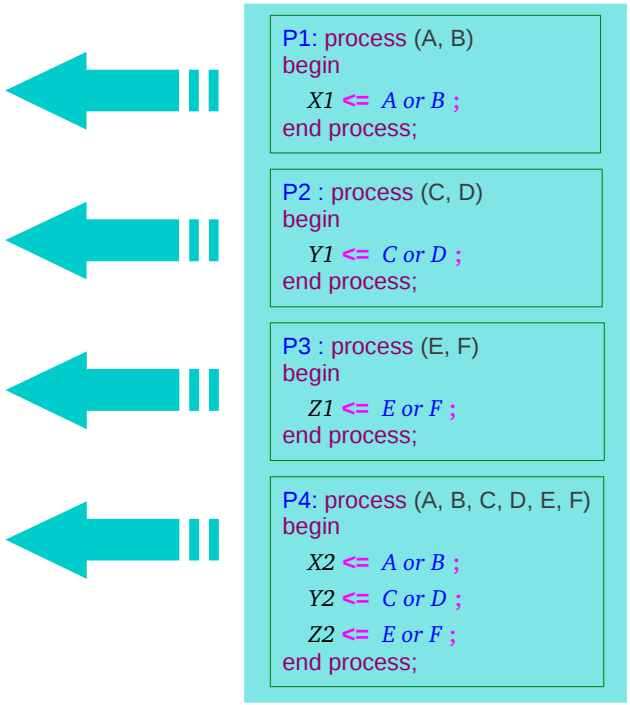
Non-deterministic Execution Order

Don't know which process executes first among P1 ~ P4.



Concurrent assignments

Sequential assignments



The order of statements is important

end

Concurrent vs Sequential (4)

Simulation of parallel activities

4 processes are executed concurrently

zero delay assignments

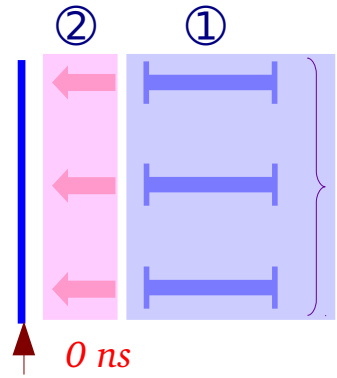
```
X1 <= A or B ;
Y1 <= C or D ;
Z1 <= E or F ;

process (A, B, C, D, E, F)
begin
    X2 <= A or B ;
    Y2 <= C or D ;
    Z2 <= E or F ;
end process;
```

only lumped view ②①

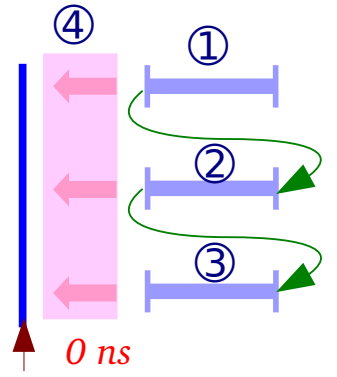
lumped view ④

* *The order of statements is important*
  1. *scheduled after some delta time*
  2. *non-blocking assignment*

Concurrent assignments



Sequential assignments

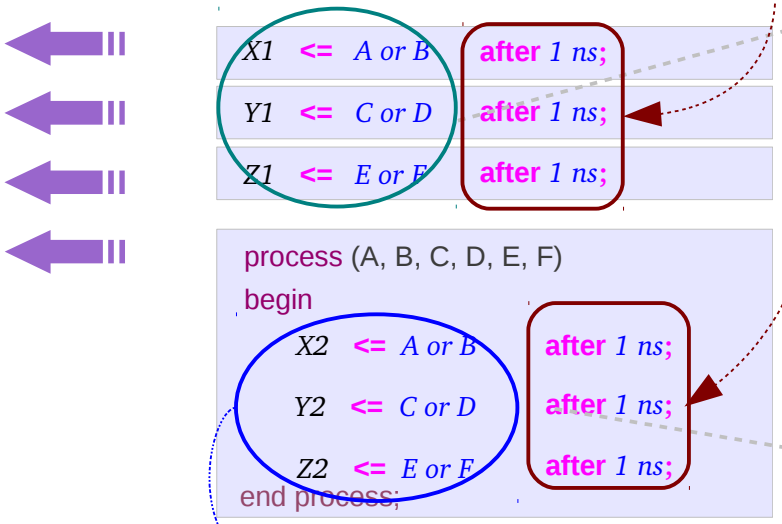


Concurrent vs Sequential (5)

Simulation of parallel activities

4 processes are executed concurrently

nonzero delay assignments



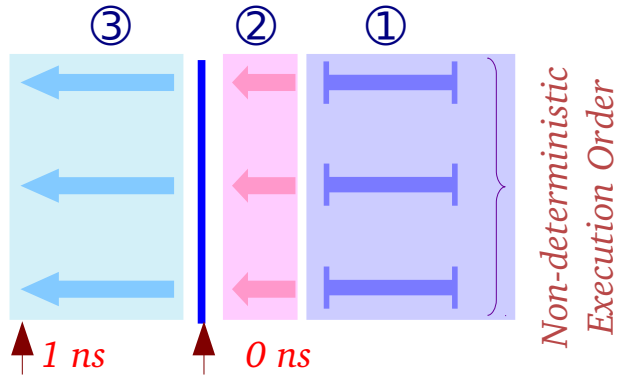
** The order of statements is important*

1. scheduled after some delta time

2. non-blocking assignment

Concurrent assignments

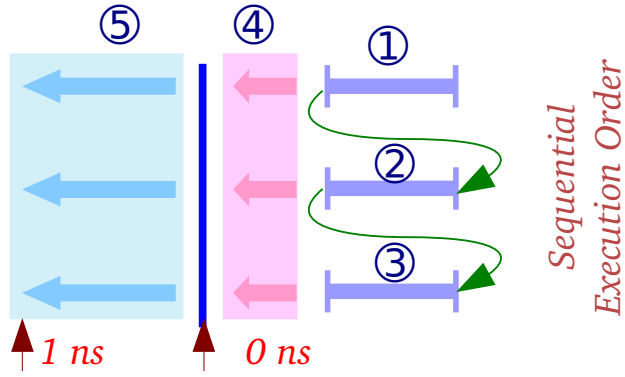
only lumped view ③②①



Scheduled Time | ***Delta Time*** | ***Evaluation***

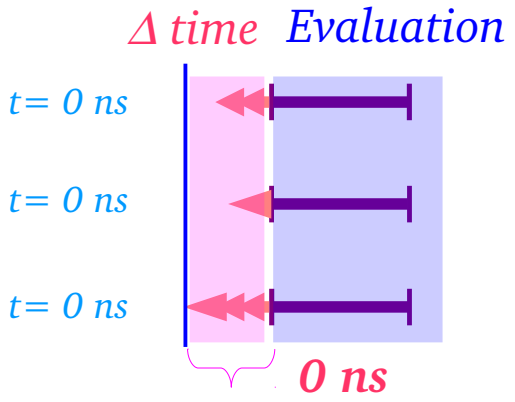
Sequential assignments

lumped view ⑤④

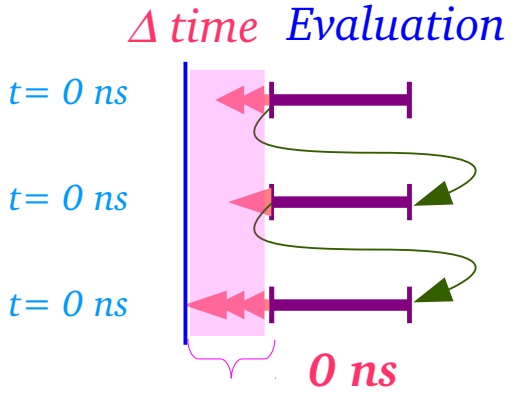


Zero Delay Assignment

```
architecture arch of entity ent is
• • •
begin
    X1 <= A or B ;
    Y1 <= C or D ;
    Z1 <= E or F ;

    process (A, B, C, D, E, F)
    begin
        X2 <= A or B ;
        Y2 <= C or D ;
        Z2 <= E or F ;
    end process;

end
```



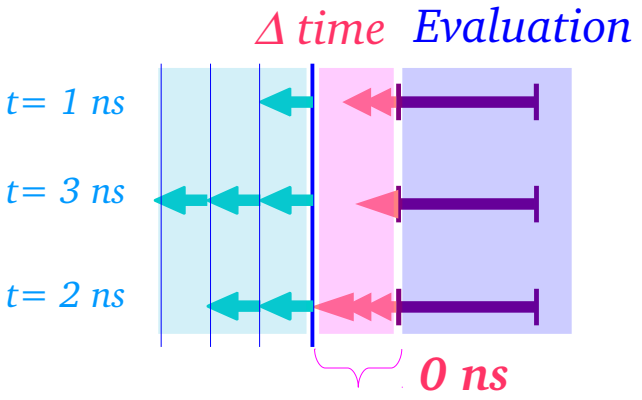
The exact no of delta is determined by the simulator and the context



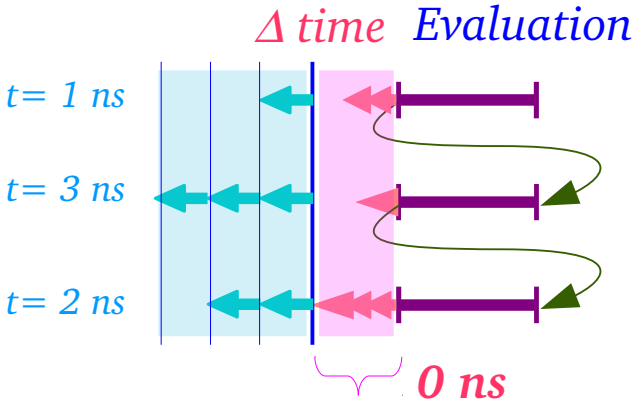
Updated values

Non-Zero Delay Assignment

```
architecture arch of entity ent is
• • •
begin
    X1 <= A or B after 1 ns;
    Y1 <= C or D after 3 ns;
    Z1 <= E or F after 2 ns;

    process (A, B, C, D, E, F)
    begin
        X2 <= A or B after 1 ns;
        Y2 <= C or D after 3 ns;
        Z2 <= E or F after 2 ns;
    end process;

end
```



The exact no of delta is determined by the simulator and the context



Not yet updated

Non-blocking Assignment (1)

architecture *arch* **of entity** *ent* **is**

• • •

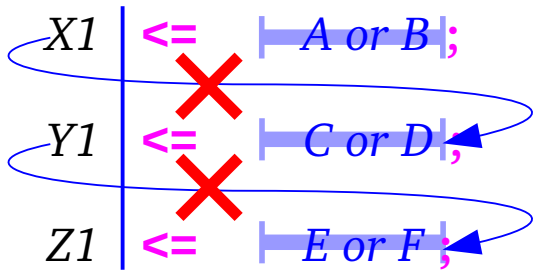
begin

```
X1 <= A or B;
Y1 <= C or D;
Z1 <= E or F;
```

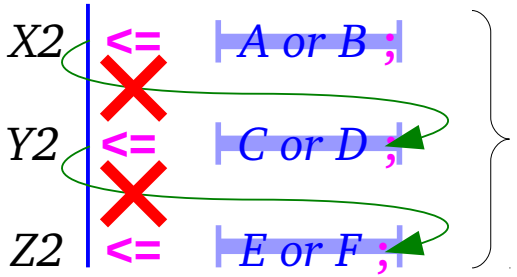
Concurrent assignment

```
process (A, B, C, D, E, F)
begin
    X2 <= A or B;
    Y2 <= C or D;
    Z2 <= E or F;
end process;
```

Sequential assignment



non-blocking assignment



end

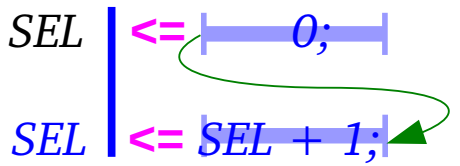
Non-blocking Assignment (2)

```
process (A, I0, I1)
begin
   SEL <=  0;
   if (A='1') then SEL <= SEL + 1; end if;
   case SEL is
        when 0
             Q <= I0;
        when 1
             Q <= I1;
   end case;
end process;
```

Scheduled on the next delta time

➡ *SEL value will not be **updated***

until the next delta time



Non-blocking Assignment

Without waiting *the next delta time,*

it can continue to process the next

sequential statement

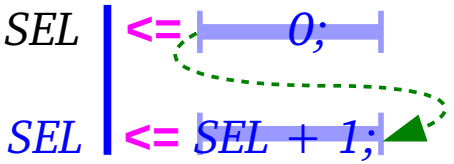
(processed with the wrong value of SEL)

Non-blocking Assignment (3)

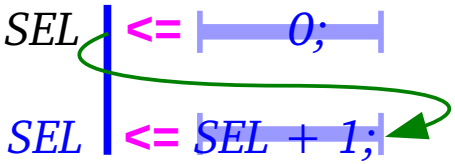
```
process
begin
    SEL <=  A or B;
    wait for 0 ns;
    if (A='1') then SEL <= SEL + 1; end if;
    wait for 0 ns;
    case SEL is
        when 0
            Q <= I0;
        when 1
            Q <= I1;
    end case;
    wait on A, I0, I1;
end process;
```

Wait for one delta time

Non-blocking
*: next statement **before** update*



wait for *0 ns;*

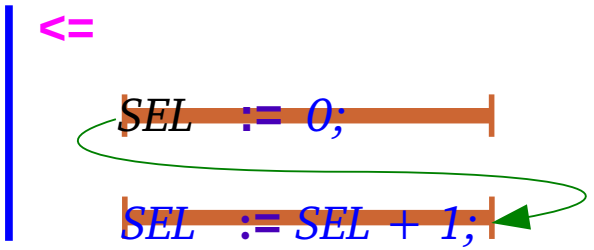


Blocking
*: next statement **after** update*

Non-blocking Assignment (4)

```
process (A, I0, I1)
  variable SEL : integer range 0 to 1;
begin
    SEL :=   A or B;
    if (A='1') then SEL := SEL + 1; end if;
    case SEL is
          when 0
                Q <= I0;
          when 1
                Q <= I1;
    end case;
end process;
```

Variable SEL changes its value immediately.



General MUX model

```
process (A, I0, I1)
begin
    case A is
        When '0'
            Q <= I0;
        When '1'
            Q <= I1;
    end case;
end process;
```

Variable Assignment (1)

```
architecture arch of entity ent is
   • • •
begin
    X1  <=      A or B   after 1 ns;
    Y1  <=      C or D   after 3 ns;
    Z1  :=      E or F   ;

    process (A, B, C, D, E, F)
        variable Y2, Z2 : bit;
    begin
        X2  <=      A or B   after 1 ns;
        Y2  :=      C or D   after 3 ns;
        Z2  :=      E or F   ;
    end process;

end
```

The Variable assignment is a sequential statement and cannot be used outside a process statement.

*The variable is **declared** here. They are used for **local storage** in process statement and subprogram body.*

The variable assignment has nothing to do with time. It executes immediately.

*It can not have an **after** clause*

Variable Assignment (2)

```
process (A, B, C, D, E, F)
    variable Z2 : bit;
begin
    X2  <=  A or B  after 1 ns;
    Y2  <=  C or D  after 3 ns;
    Z2  :=  E or F  ;
end process;
```

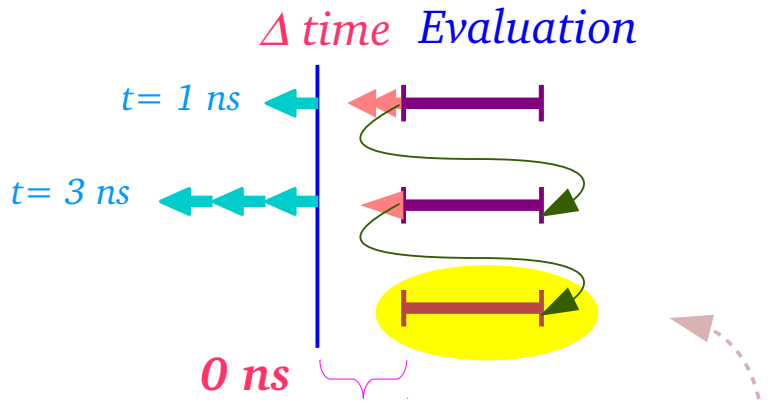
```
X2  <=  A or B  after 1 ns;
Y2  <=  C or D  after 3 ns;
Z2  :=  E or F  ;
```

```
process (A, B, C, D, E, F)
    variable Y2 : bit;
begin
    X2  <=  A or B  after 1 ns;
    Y2  :=  C or D  ;
    Z2  <=  E or F  after 2 ns;
end process;
```

```
X2  <=  A or B  after 1 ns;
Y2  :=  C or D  ;
Z2  <=  E or F  after 2 ns;
```

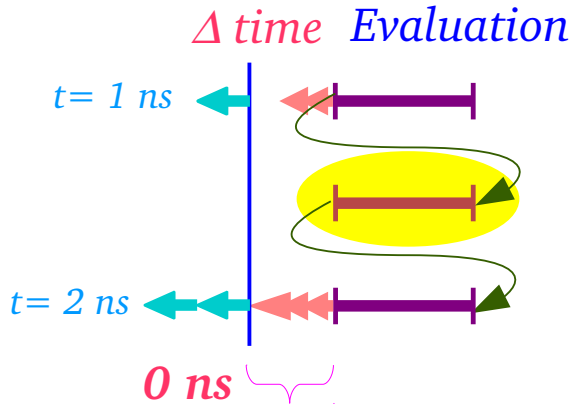
Variable Assignment (3)

```
process (A, B, C, D, E, F)
    variable Z2 : bit;
begin
    X2  <=  A or B  after 1 ns;
    Y2  <=  C or D  after 3 ns;
    Z2  :=  E or F  ;
end process;
```



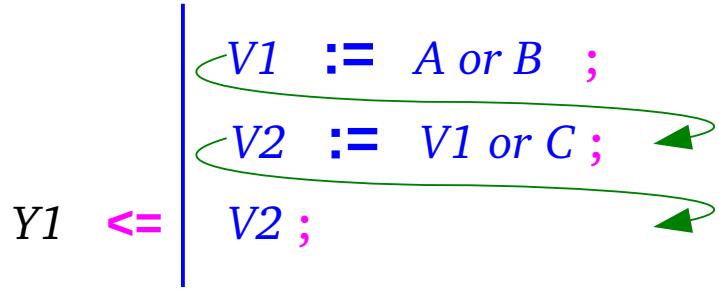
The variable assignment has nothing to do with time. It executes immediately.

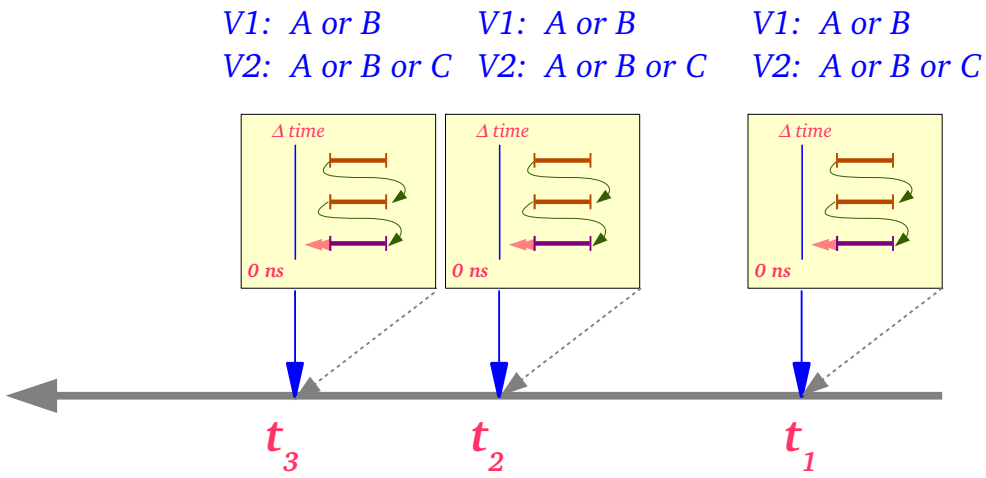
```
process (A, B, C, D, E, F)
    variable Y2 : bit;
begin
    X2  <=  A or B  after 1 ns;
    Y2  :=  C or D  ;
    Z2  <=  E or F  after 2 ns;
end process;
```



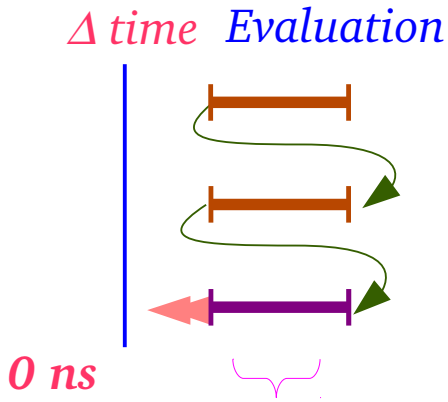
Mixed Assignments Example (1)

```
process (A, B, C)
    variable V1, V2 : bit;
begin
    V1  :=  A or B  ;
    V2  :=  V1 or C ;
    Y1  <=  V2 ;
end process;
```





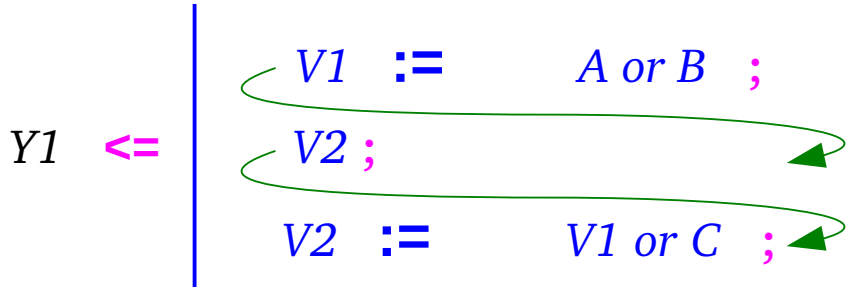
Times when A or B or C changes

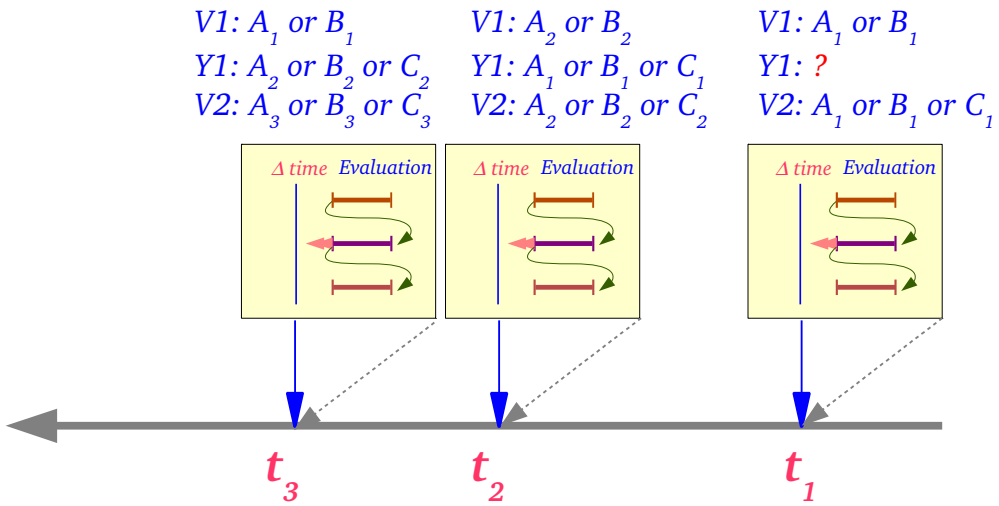


Y1 <= A or B or C ;

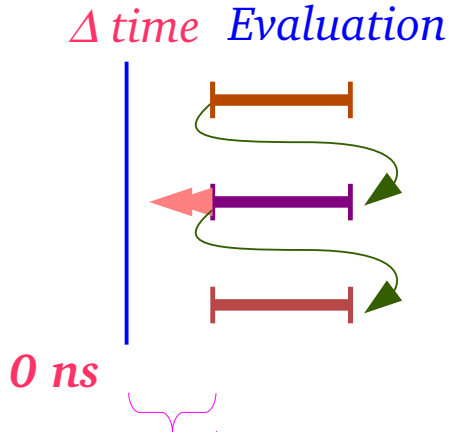
Mixed Assignments Example (2)

```
process (A, B, C)
    variable V1, V2 : bit;
begin
    V1  :=  A or B  ;
    Y1  <=  V2 ;
    V2  :=  V1 or C  ;
end process;
```





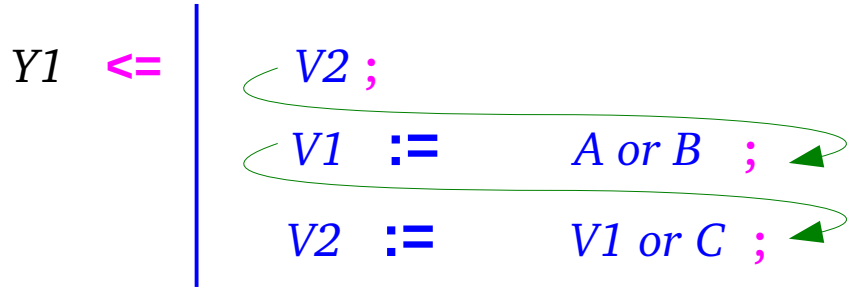
Times when A or B or C changes

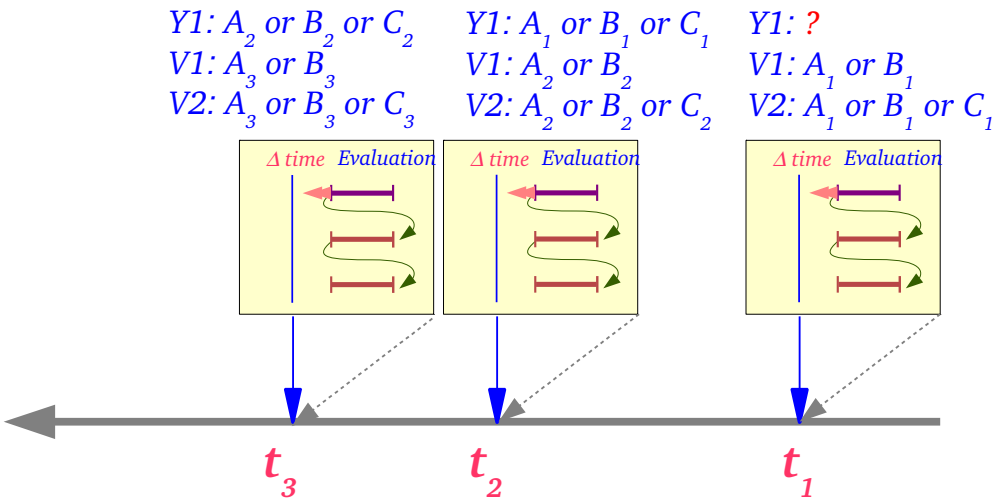


Y1 <= A or B or C ;

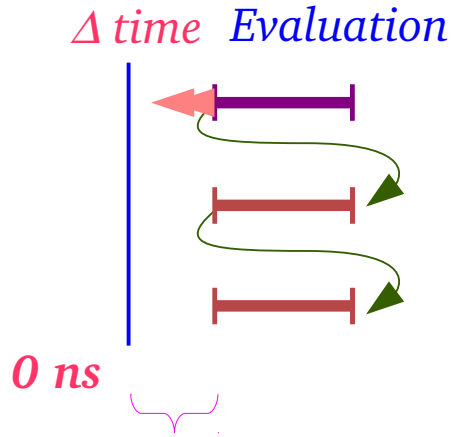
Mixed Assignments Example (3)

```
process (A, B, C)
    variable V1, V2 : bit;
begin
    Y1  <=  V2 ;
    V1  :=  A or B ;
    V2  :=  V1 or C ;
end process;
```





Times when A or B or C changes



Y1 <= A or B or C ;

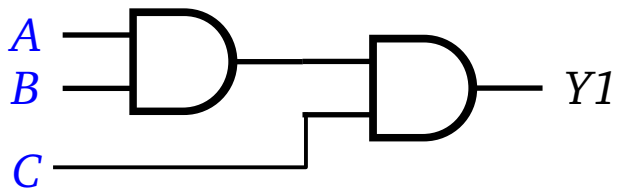
Mixed Assignments Example (4)

```
process (A, B, C)
    variable V1, V2 : bit;
begin
    V1  :=  A or B  ;
    V2  :=  V1 or C ;
    Y1  <=  V2 ;
end process;
```

```
process (A, B, C)
    variable V1, V2 : bit;
begin
    Y1  <=  V2 ;
    V1  :=  A or B  ;
    V2  :=  V1 or C ;
end process;
```

```
process (A, B, C)
    variable V1, V2 : bit;
begin
    V1  :=  A or B  ;
    Y1  <=  V2 ;
    V2  :=  V1 or C ;
end process;
```

Same Synthesis Result



References

[1] http://en.wikipedia.org/
[2] J. V. Spiegel, VHDL Tutorial,
http://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html
[3] J. R. Armstrong, F. G. Gray, Structured Logic Design with VHDL
[4] Z. Navabi, VHDL Analysis and Modeling of Digital Systems
[5] D. Smith, HDL Chip Design
[6] http://www.csee.umbc.edu/portal/help/VHDL/stdpkg.html
[7] VHDL Tutorial - VHDL onlinewww.vhdl-online.de/tutorial/