

# Signals & Variables

---

Copyright (c) 2012 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using OpenOffice and Octave.

# 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**

# Sequential Statement

- Wait Statement
- Assertion Statement
- Report Statement
- Generate Statement
- Signal Assignment
- Variable Assignment
- Procedure Call
- If
- Case
- Loop
- Next
- Exit
- Return
- Null

- **Case Statement**
- **If Statement**
- **Loop Statement**
- **Process Statement**
- **Subprogram Body**

- Sequential Signal Assignment

- Conditional Signal Assignment
- Selected Signal Assignment

X

# Conditional Signal Assignment

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

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

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

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

← *simple concurrent statement*

• Concurrent Signal Assignment

- Conditional Signal Assignment
- Selected Signal Assignment

# Selected Signal Assignment

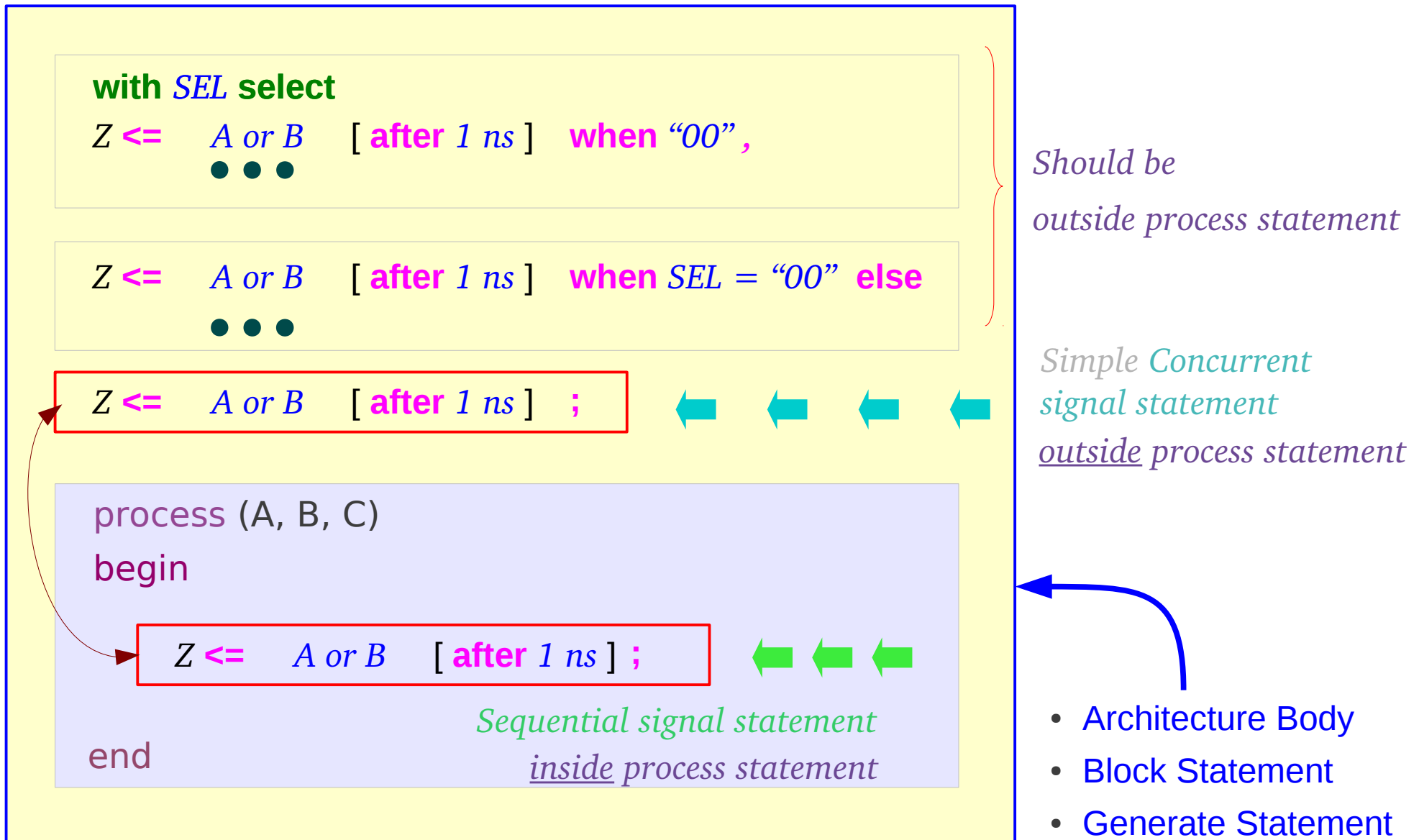
- Conditional Signal Assignment

```
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 ]  ;
```

- 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;
```

# Concurrent vs Sequential



# Order of Statements

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 ;

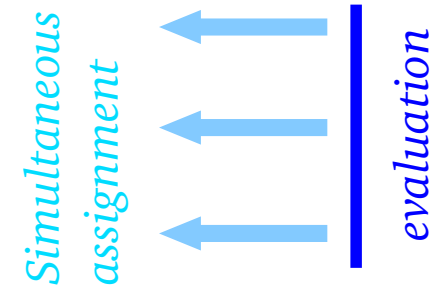
Y2 ← C or D ;

Z2 ← E or F ;

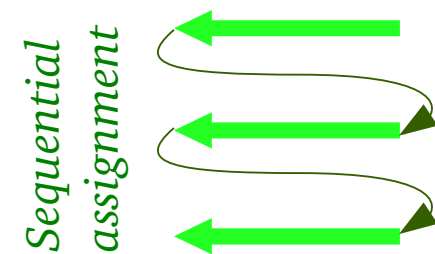
end

end

*Simulation of parallel activities*

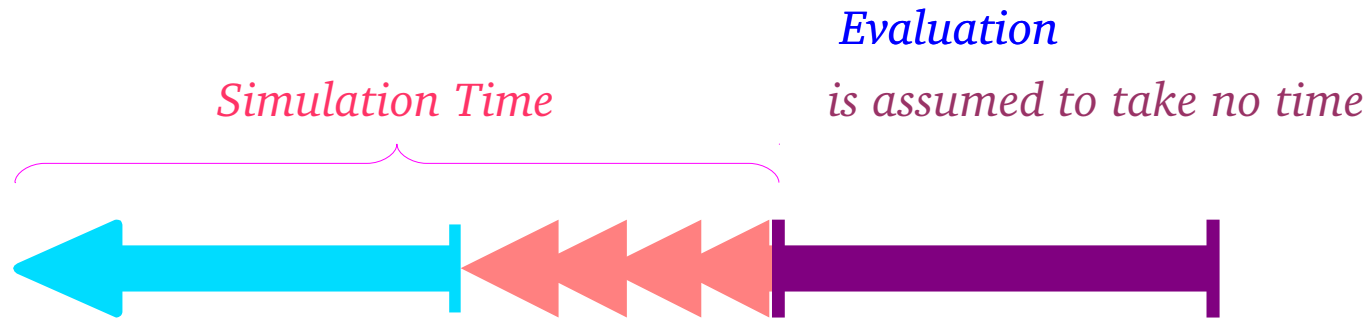


*The order of statements is important*





# Simulation Time



Unit: *ms, ns, ps, ...*

*Unitless Delta  $\Delta$*

***Real Delay***

– used for a simulator to  
mimic parallel activities  
*simulator*

$$1 \text{ ms} = 1000 \text{ ns}$$

$$1 \text{ ns} = 1000 \text{ ps}$$

$$1 \text{ ps} \neq n \cdot \Delta$$

*no integer  $n$  that make  $n$  delta  
equal to 1 ps.*

$$n \cdot \Delta = 0 \text{ ps} = 0 \text{ ns} \dots$$

***Zero Delay***

***Zero Delay Assignment***

```
X1 <= A or B ;
```

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

# 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

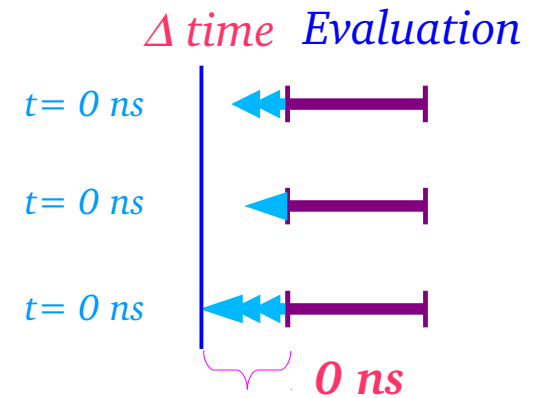
X2 ← A or B ;

Y2 ← C or D ;

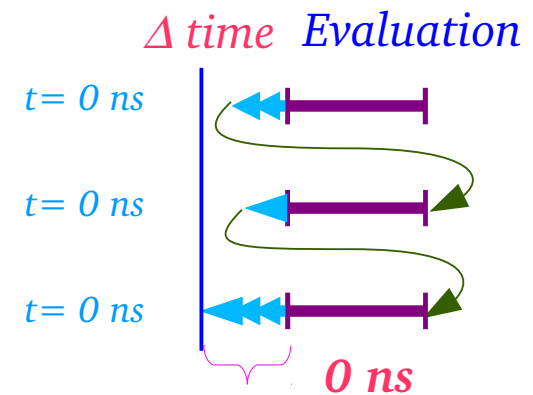
Z2 ← E or F ;

end

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  $\leftarrow$  A or B after 1 ns;

Y1  $\leftarrow$  C or D after 3 ns;

Z1  $\leftarrow$  E or F after 2 ns;

process (A, B, C, D, E, F)

begin

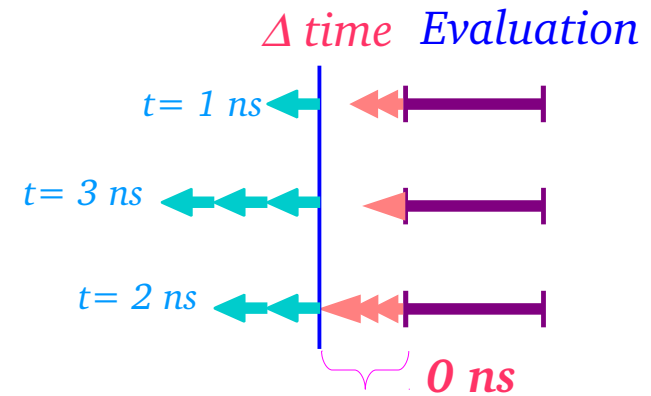
X2  $\leftarrow$  A or B after 1 ns;

Y2  $\leftarrow$  C or D after 3 ns;

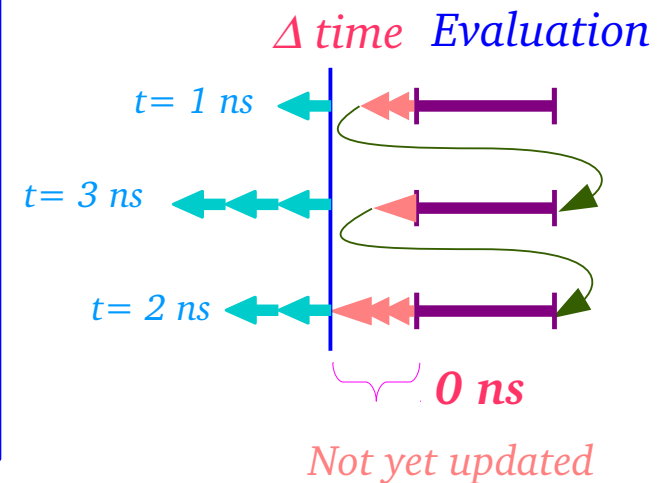
Z2  $\leftarrow$  E or F after 2 ns;

end

end



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



# 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
```

```
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 (A, B, C, D, E, F)
```

```
  variable Y2 : bit;
```

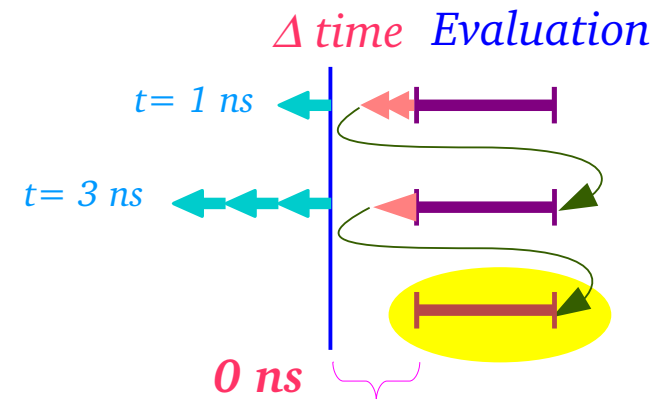
```
begin
```

```
  X2 <= A or B after 1 ns;
```

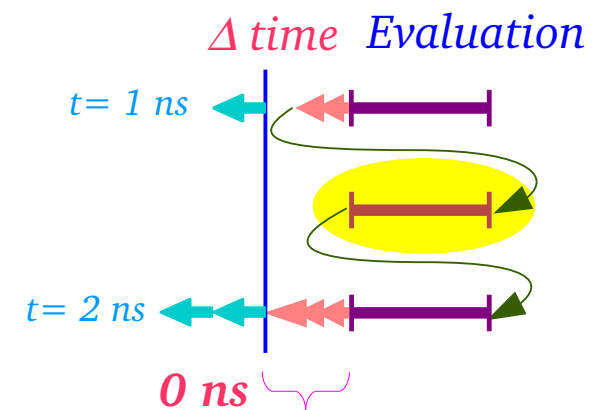
```
  Y2 := C or D ;
```

```
  Z2 <= E or F after 2 ns;
```

```
end
```



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



## References

[1] <http://en.wikipedia.org/>

[2]