Signals & Variables

Young Won Lim
03/20/2012

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

This document was produced by using OpenOffice and Octave.

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

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

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** → **condition**

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"* , → **selection**

A or E [**after** *4 ns*] **when** *"11"* ,

A or F [**after** *5 ns*] **when others;**

Concurrent vs Sequential

with *SEL* **select**   *condition*

Z **<=**   *A or B*   [**after** *1 ns*]   **when** *"00"* ,

● ● ●

Z **<=**   *A or B*   [**after** *1 ns*]   **when** *SEL = "00"* **else**

● ● ●   *selection*

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

Outside process statement

Simple Concurrent
signal statement
outside *process*
statement

process (A, B, C)

begin

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

Sequential signal statement
inside *process statement*

end process

- Architecture Body
- Block Statement
- Generate Statement

Young Won Lim
03/20/2012

Order of Execution (1)

```
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
```

Simulation of parallel activities

Simultaneous assignment

Non-deterministic Execution Order

0 ns

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

Sequential assignment

Sequential Execution Order

0 ns

The order of statements is important

Order of Execution (2)

architecture *arch* of entity *ent* is
● ● ●
begin

X1 <= *A or B* ;

Y1 <= *C or D* ;

Z1 <= *E or F* ;

Non-deterministic Execution Order

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

begin

X2 <= *A or B* ;

Y2 <= *C or D* ;

Z2 <= *E or F* ;

end process;

end

Simulation of parallel activities

Simultaneous assignment

Non-deterministic Execution Order

0 ns

Scheduled Time   Delta Time   Evaluation

Sequential assignment

Sequential Execution Order

0 ns

The order of statements is important

Order of Execution (3)

architecture *arch* of entity *ent* is
• • •
begin

X1 <= A or B after 1 ns;

Y1 <= C or D after 1 ns;

Z1 <= E or F after 1 ns;

process (A, B, C, D, E, F)
begin
X2 <= A or B after 1 ns;

Y2 <= C or D after 1 ns;

Z2 <= E or F after 1 ns;
end process;

end

Simulation of parallel activities

Simultaneous assignment

Non-deterministic Execution Order

1 ns   0 ns

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

Sequential assignment

Sequential Execution Order

1 ns   0 ns
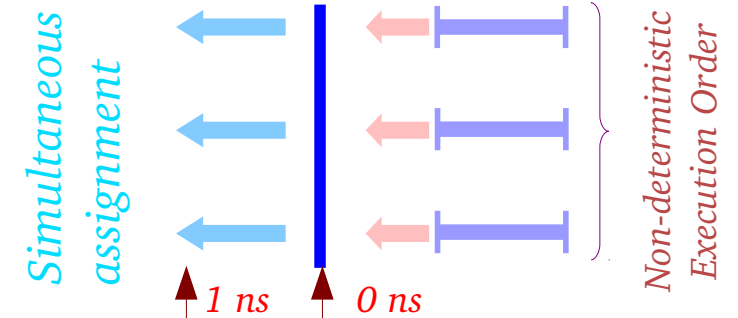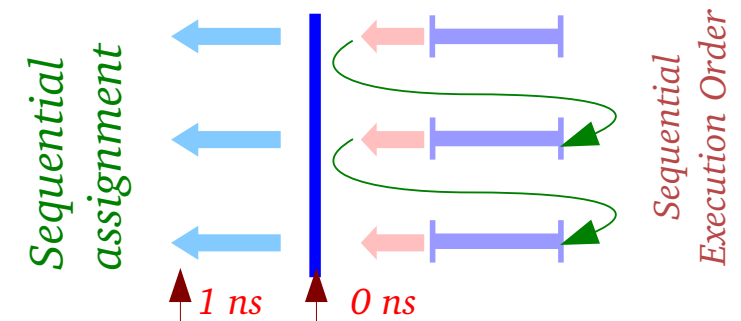
The order of statements is important

Non-blocking Assignment

Simulation Time

Evaluation is assumed to take no time

Simulation Time

Unit: ms, ns, ps, …
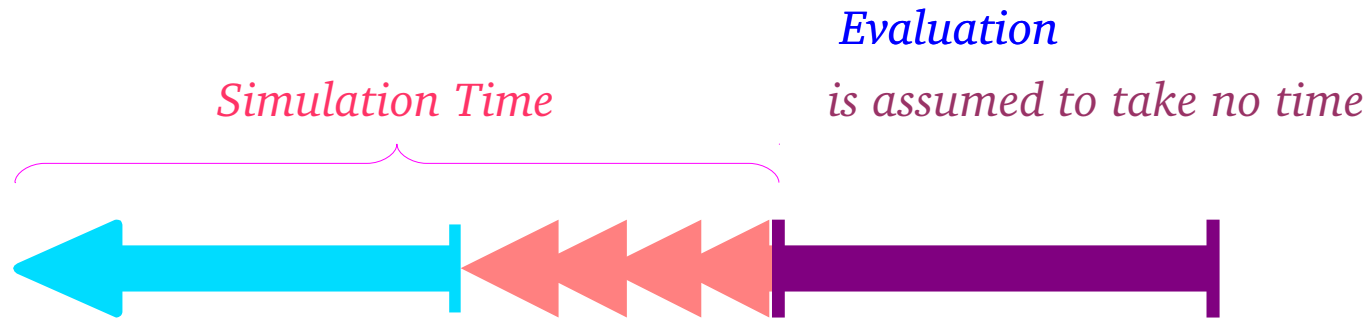
Real Delay

$$1\,ms \;=\; 1000\,ns$$
$$1\,ns \;=\; 1000\,ps$$

Unitless Delta Δ

– used for a simulator to mimic parallel activities simulator

$$1\,ps \;\ne\; n{\cdot}\Delta$$

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

$$n{\cdot}\Delta \;=\; 0\,ps \;=\; 0\,ns \;\cdots$$

Zero Delay

Zero Delay Assignment

$$X1 \;\text{<=}\; A \text{ or } B \;;$$

$$X1 \;\text{<=}\; A \text{ or } B \;\textbf{after}\; 0\,ns;$$

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 process;

end
```


Δ time   Evaluation

t= 0 ns

t= 0 ns

t= 0 ns

0 ns

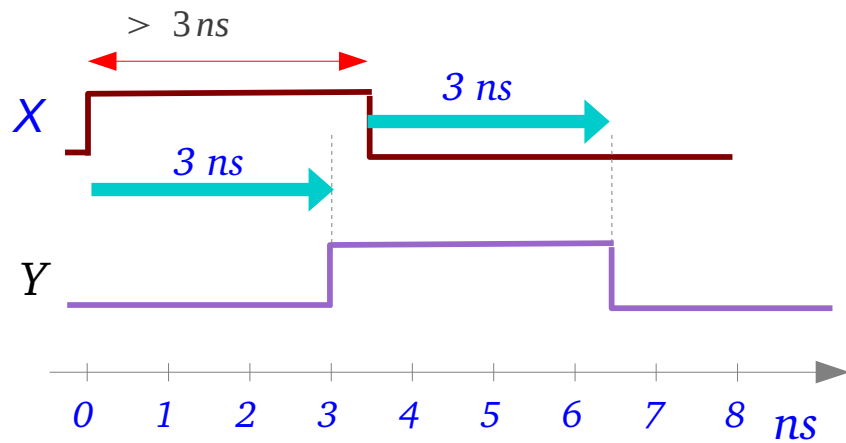*The exact no of delta is determined by the simulator and the context*
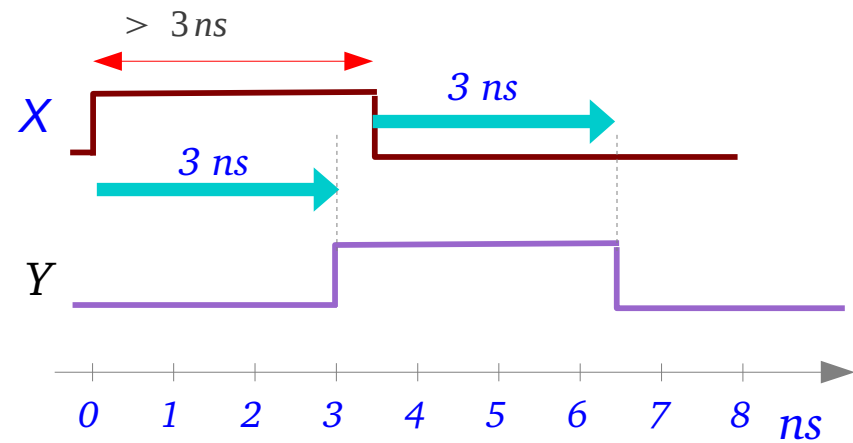
Δ time   Evaluation

t= 0 ns

t= 0 ns

t= 0 ns

0 ns
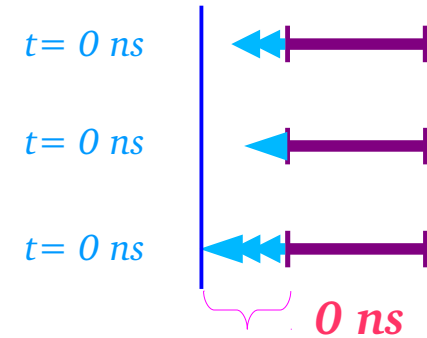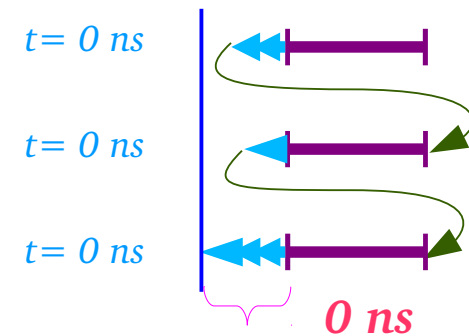
Updated values

Non-Zero Delay Assignment
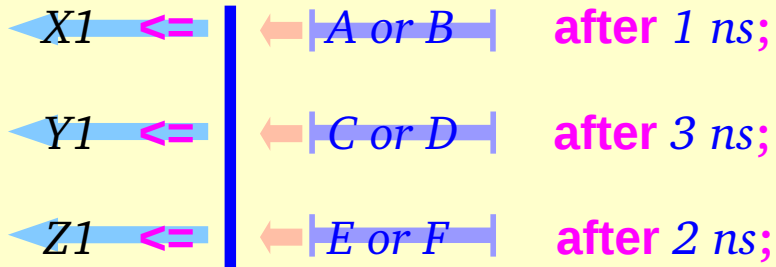
architecture *arch* **of entity** *ent* **is**

● ● ●

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** 1 ns;

Z2 **<=** E or F **after** 1 ns;

end process;

end

Δ *time* *Evaluation*

t= 1 ns

t= 3 ns

t= 2 ns

0 ns

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

Δ *time* *Evaluation*

t= 1 ns

t= 3 ns

t= 2 ns

0 ns

Not yet updated
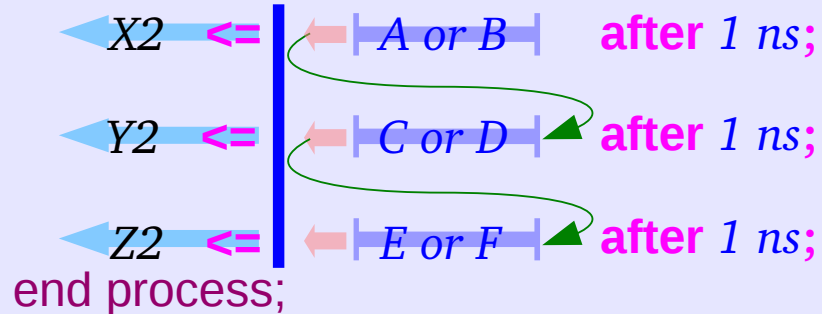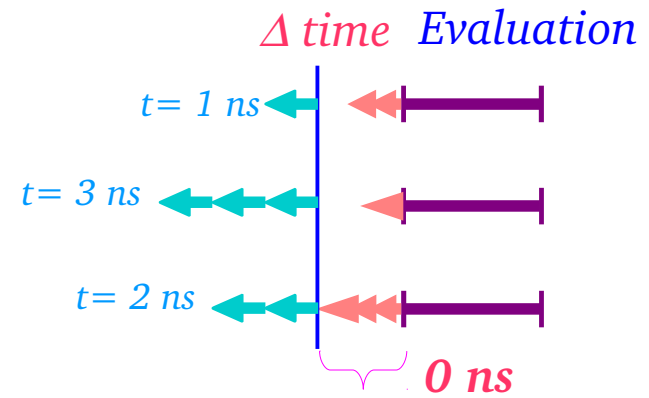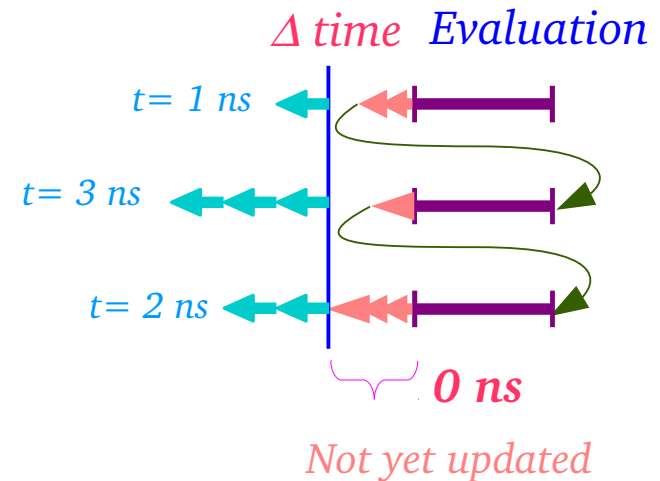
Inertial Delay & Transport Delay

$Y \quad \textbf{<=} \quad X \quad \textbf{after } 3\,ns;$

$Y \quad \textbf{<=} \quad \textbf{transport } X \quad \textbf{after } 3\,ns;$

Signals & Variables

Young Won Lim
03/20/2012
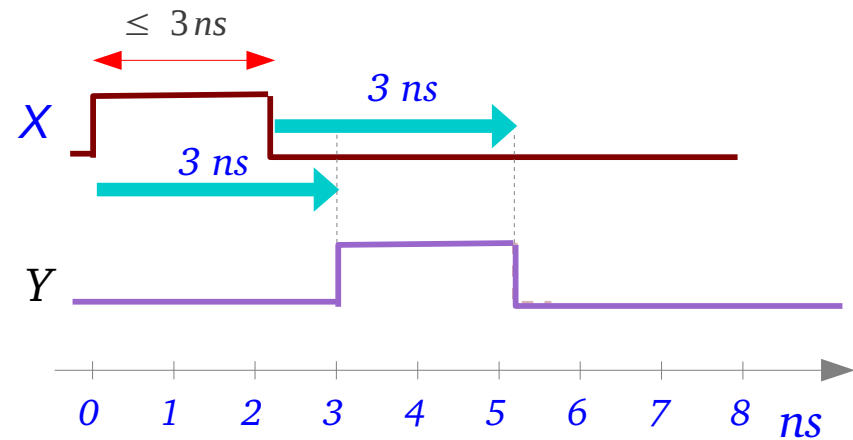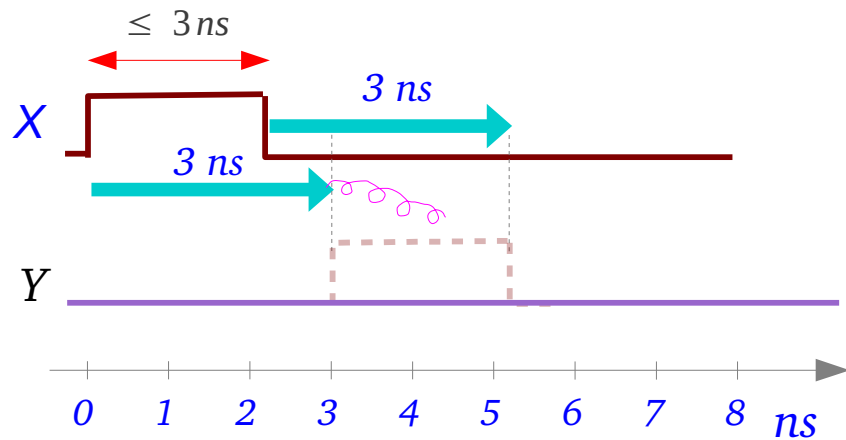
Variable Assignment (1)

```
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 <u>sequential statement</u> 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;
```
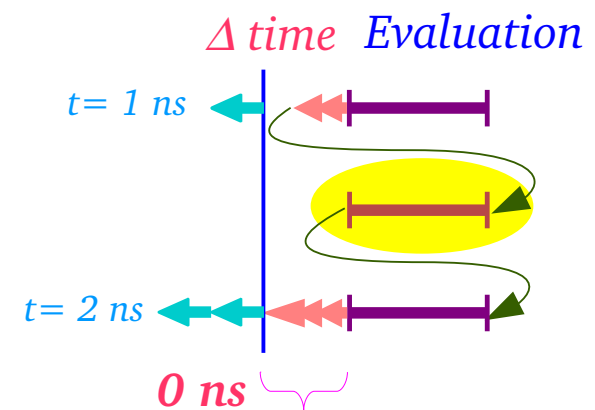

The variable assignment has nothing to do with time. It executes immediately.

```
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;
```
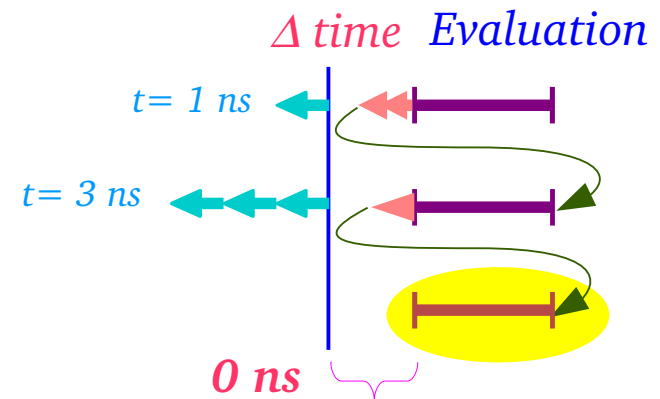
Multiple Assignments to the Same Target

```
architecture arch of entity ent is
    • • •
begin

    X1  <=        A or B ;
    X1  <=        C or D ;

    process (A, B, C, D, E, F)
        variable Z2 : bit;
    begin

        Y2  <=        A or B ;
        Y2  <=        C or D ;


        Z2  :=        A or B ;
        Z2  :=        C or D ;


    end process;

end
```

Multiple Concurrent Assignments

Multiple Concurrent Assignment is *legal* *only when a* *resolution* *function* *is defined.*

(wire-and, wire-or)

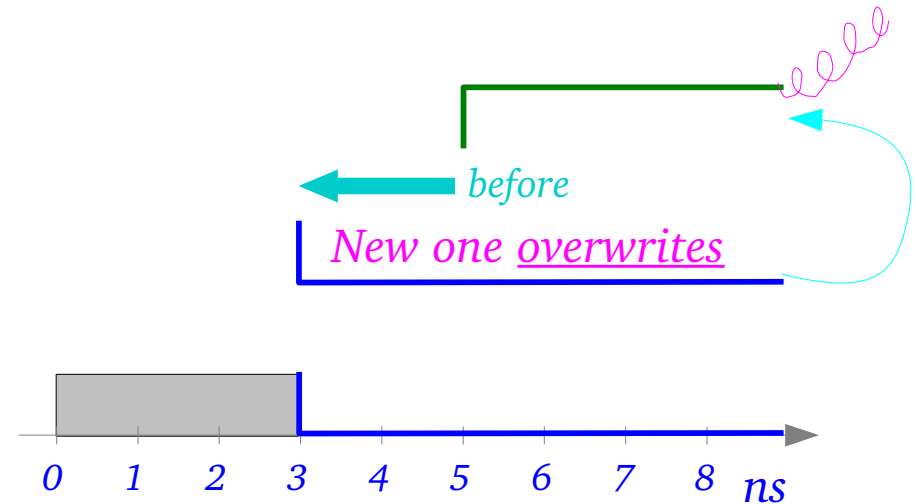**Multiple Sequential Assignments**

- *Overwrite*
- *Append*
- *Keep*

Multiple Variable Assignments

Variable Z2 has the result of the latest assignments (The new assignment overwrites the old one)
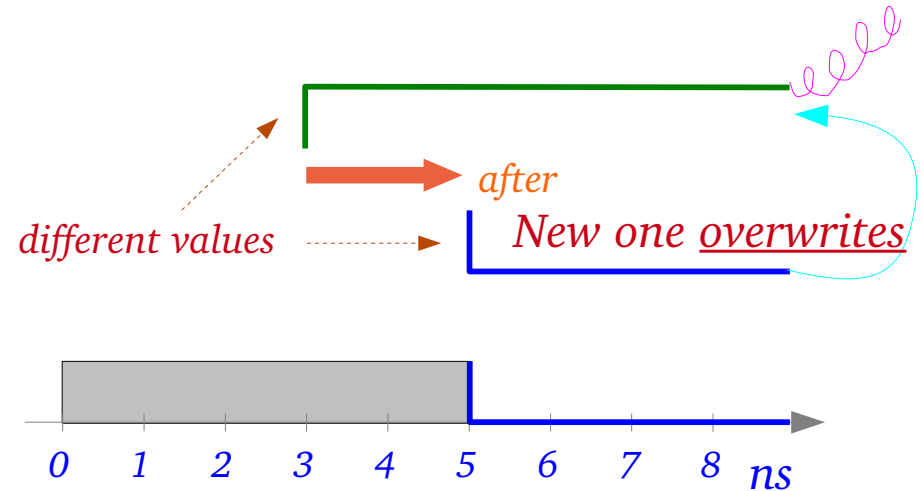
Multiple Sequential Assignments

```
architecture arch of entity ent is
    • • •
begin

    X1  <=          A or B ;
    X1  <=          C or D ;

    process (A, B, C, D, E, F)
        variable Z2 : bit;
    begin

        Y2  <=          A or B ;
        Y2  <=          C or D ;


        Z2  :=          A or B ;
        Z2  :=          C or D ;


    end process;

end
```

Multiple Concurrent Assignments

Multiple Concurrent Assignment is legal only when a resolution function is defined.

(wire-and, wire-or)

Multiple Sequential Assignments

- *Overwrite*
- *Append*
- *Keep*

Multiple Variable Assignments

Variable Z2 has the result of the latest assignments (The new assignment overwrites the old one)

Inertial Delay (1)

Multiple Sequential Assignments

process (…)

begin

    X2 **<=**   *'1'* **after** *5 ns***;**

    X2 **<=**   *'0'* **after** *3 ns***;**

end process;

before

New one overwrites

0 1 2 3 4 5 6 7 8 *ns*

process (…)

begin

    X2 **<=**   *'1'* **after** *3 ns***;**

    X2 **<=**   *'0'* **after** *5 ns***;**

end process;

after

different values

New one overwrites

0 1 2 3 4 5 6 7 8 *ns*

Inertial Delay (2)

Multiple Sequential Assignments

process (…)

begin

      X2 **<=**   '1' **after** *5 ns*;

      X2 **<=**   '1' **after** *3 ns*;

end process;

before

New one overwrites
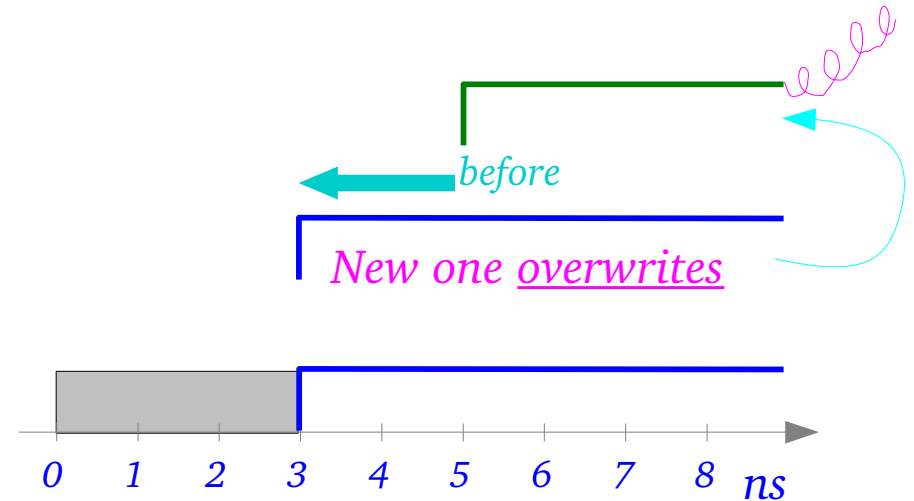
process (…)

begin

      X2 **<=**   '0' **after** *3 ns*;

      X2 **<=**   '0' **after** *5 ns*;

end process;

after

same value

Both are kept
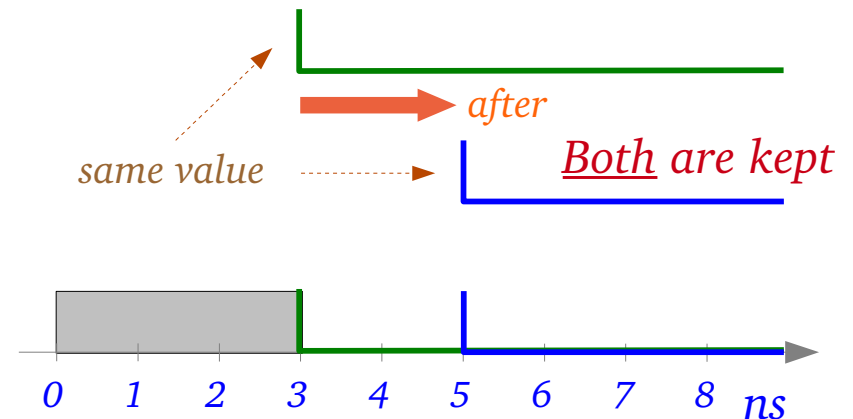
Transport Delay (1)

Multiple Sequential Assignments

```
process (…)
begin

    X2   <= transport  '1'  after 5 ns;
    X2   <= transport  '0'  after 3 ns;

end process;
```
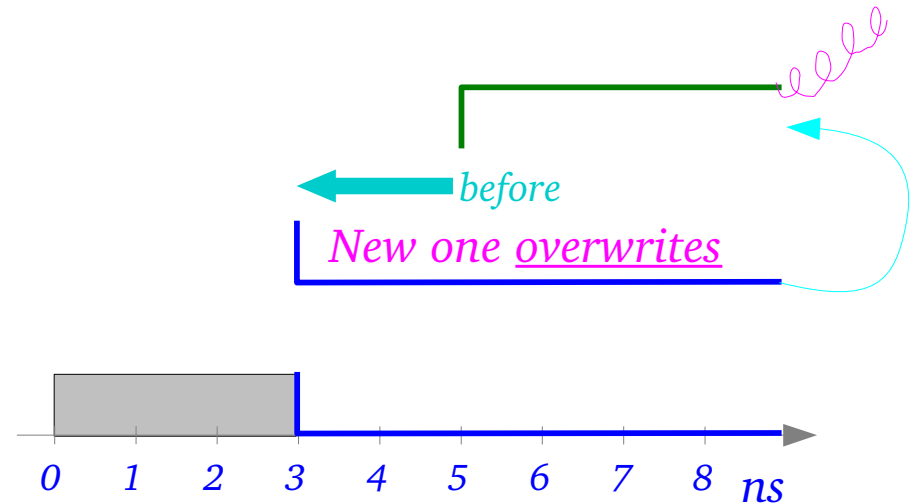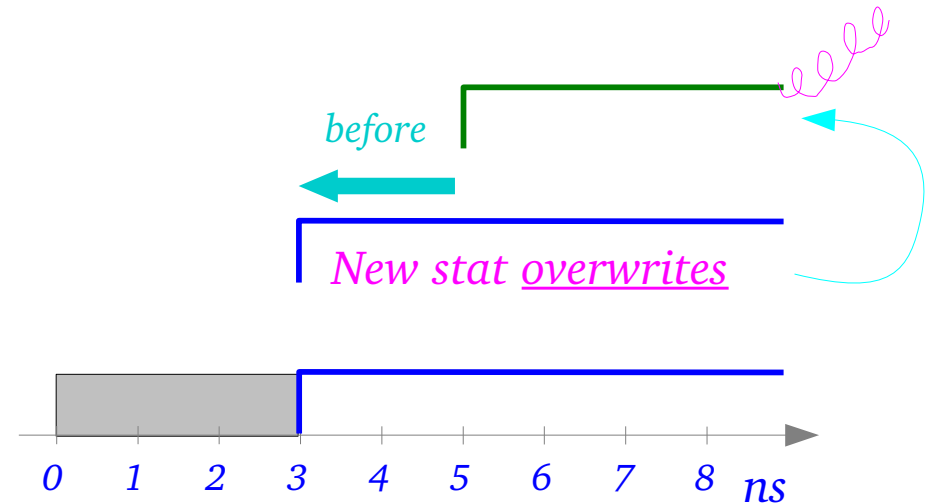
before

New one overwrites

0 1 2 3 4 5 6 7 8 *ns*

```
process (…)
begin

    X2   <= transport  '1'  after 3 ns;
    X2   <= transport  '0'  after 5 ns;

end process;
```

after

New one is appended

0 1 2 3 4 5 6 7 8 *ns*

Young Won Lim
03/20/2012
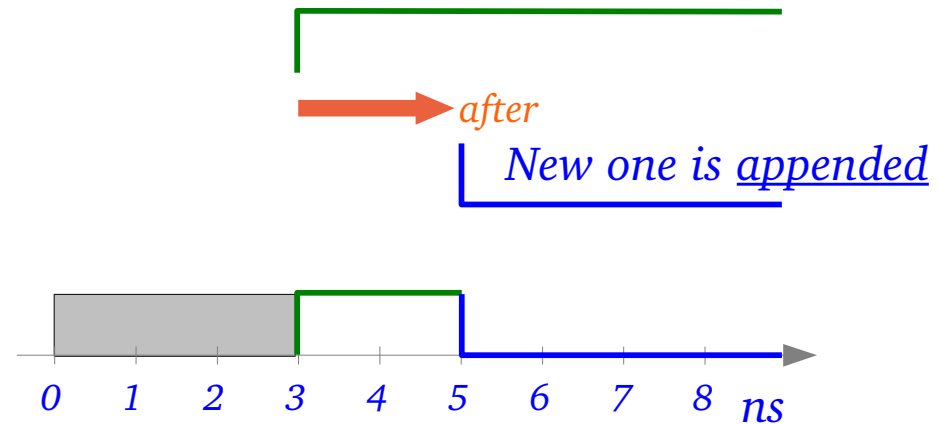
Transport Delay (2)

Multiple Sequential Assignments

```
process (…)
begin

    X2  <= transport  '1'  after 5 ns;
    X2  <= transport  '1'  after 3 ns;

end process;
```

before

New stat overwrites

```
0   1   2   3   4   5   6   7   8   ns
```

```
process (…)
begin

    X2  <= transport  '0'  after 3 ns;
    X2  <= transport  '0'  after 5 ns;

end process;
```

after

New one is appended

```
0   1   2   3   4   5   6   7   8   ns
```
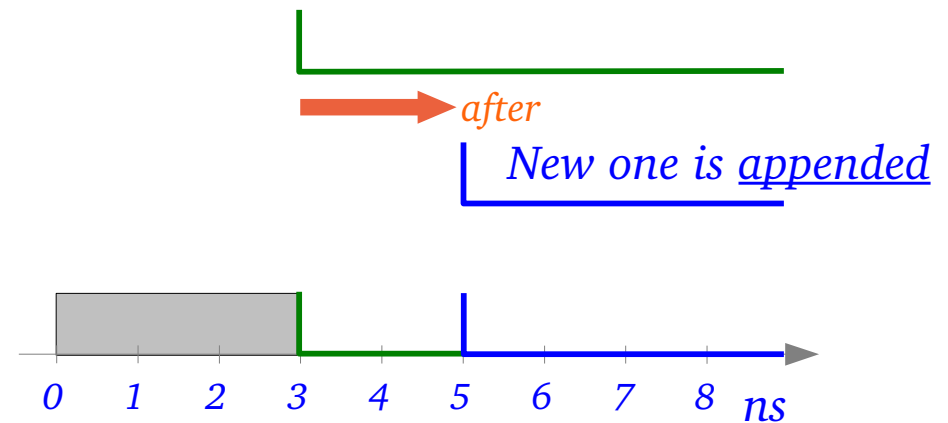
Inertial Delay

Multiple Sequential Assignments – Inertial Delay

```
process (…)
begin

    X2   <=        v1   after t1 ns;
    X2   <=        v2   after t2 ns;

end process;
```

$t2 < t1$ $v2 = v1$ *New one overwrites*

$v2 \neq v1$ *New one overwrites*
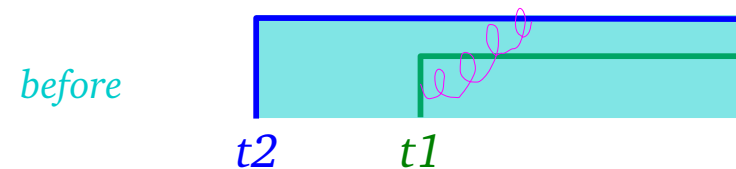
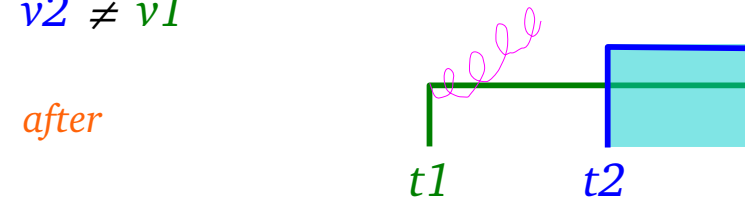$t1 < t2$ $v1 = v2$ *Both are kept*

$v1 \neq v2$ *New one overwrites*

before
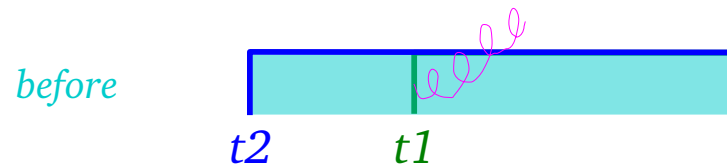
t2   *t1*

$t2 < t1$ *New one overwrites*
$v2 = v1$

after

t1   *t2*

$t1 < t2$ *Both are kept*
$v1 = v2$

before

t2   *t1*

$t2 < t1$ *New one overwrites*
$v2 \neq v1$

after

t1   *t2*

$t1 < t2$ *New one overwrites*
$v1 \neq v2$

Transport Delay

Multiple Sequential Assignments – Transport Delay

process (…)
begin

     X2 **<= transport** *v1* **after** *t1 ns***;**
     X2 **<= transport** *v2* **after** *t2 ns***;**

end process;

$t2 < t1$ *New stat <u>overwrites</u>*

$t1 < t2$ *New stat is <u>appended</u>*

before
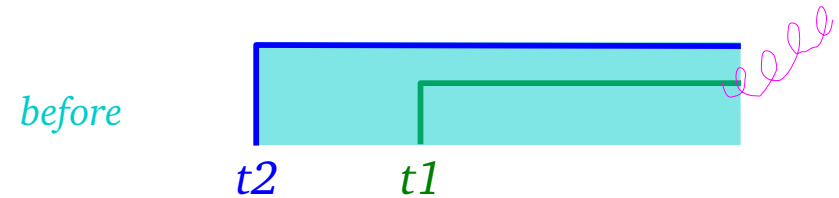
    t2   *t1*

$t2 < t1$ *New one overwrites*
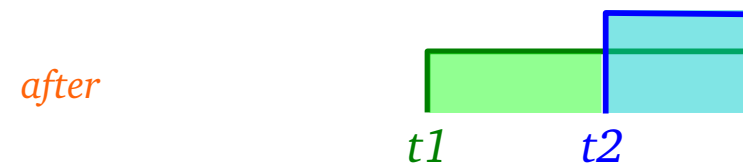
after

    t1   *t2*

$t1 < t2$ *New one is appended*

Inertial & Transport Delay Model (1)

Inertial Delay

The simulation time of a new event		
Before the time of an old one		
	New one <u>overwrites</u>	
After the time of an old one		
	For the **same** value	
		Both are kept
	For **different** values	
		New one overwrites

t2 < t1	v2 = v1	New one overwrites
	v2 ≠ v1	New one overwrites
t1 < t2	v1 = v2	Both are kept
	v1 ≠ v2	New one overwrites

Transport Delay

The simulation time of a new event	
Before the time of an old one	
	New one <u>overwrites</u>
After the time of an old one	
	New stat is <u>appended</u>

t2 < t1	New stat <u>overwrites</u>
t1 < t2	New stat is <u>appended</u>

Young Won Lim
03/20/2012

Inertial & Transport Delay Model (2)

Inertial Delay

The simulation time of a new event		
Before the time of an old one		
	New one <u>overwrites</u>	
After the time of an old one		
	For the **same** value	
		Both are kept
	For **different** values	
		New one overwrites

$t2 < t1$	$v2 = v1$	New one overwrites
	$v2 \neq v1$	New one overwrites
$t1 < t2$	$v1 = v2$	Both are kept
	$v1 \neq v2$	New one overwrites

$$Y \quad \textbf{<=} \quad X \quad \textbf{after } 3 \, ns;$$

Young Won Lim
03/20/2012
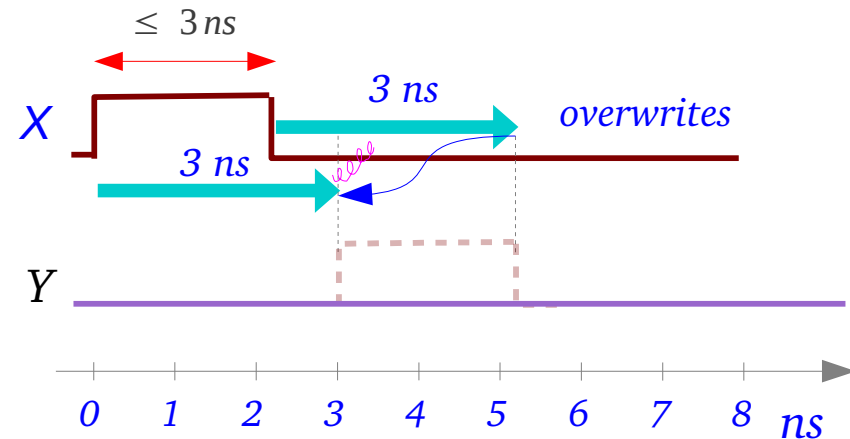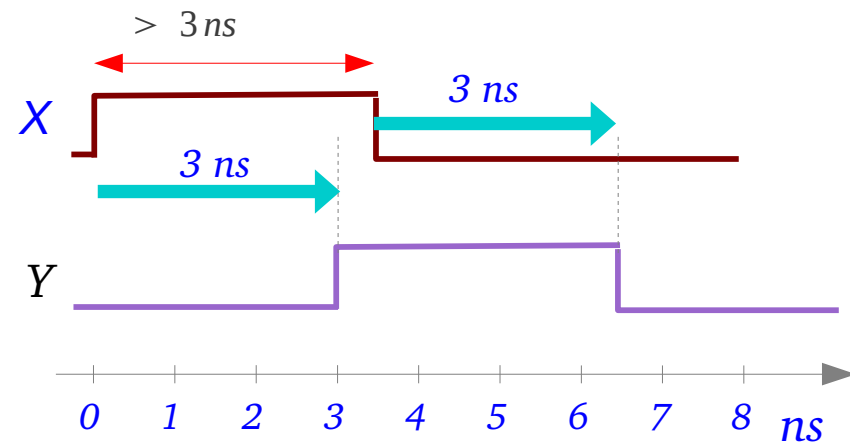
Inertial & Transport Delay Model (3)

Transport Delay

The simulation time of a new event
Before the time of an old one
New one <u>overwrites</u>
After the time of an old one
New stat is <u>appended</u>

t2 < t1	*New stat* <u>overwrites</u>
t1 < t2	*New stat is* <u>appended</u>

Y **<=**   **transport** X **after** *3 ns;*
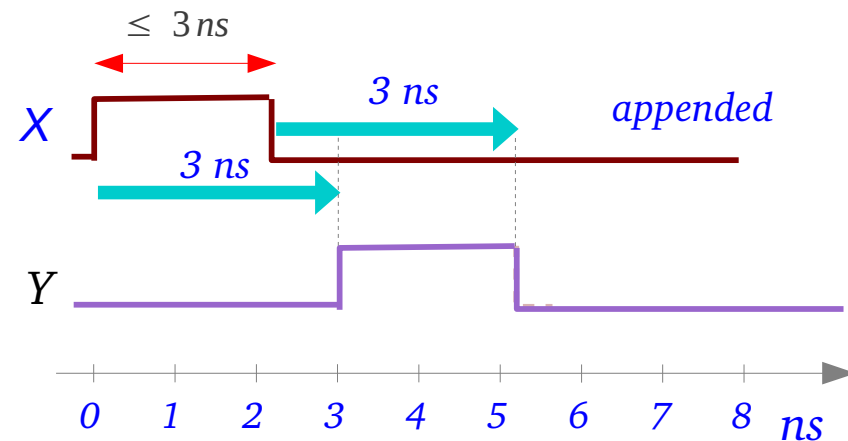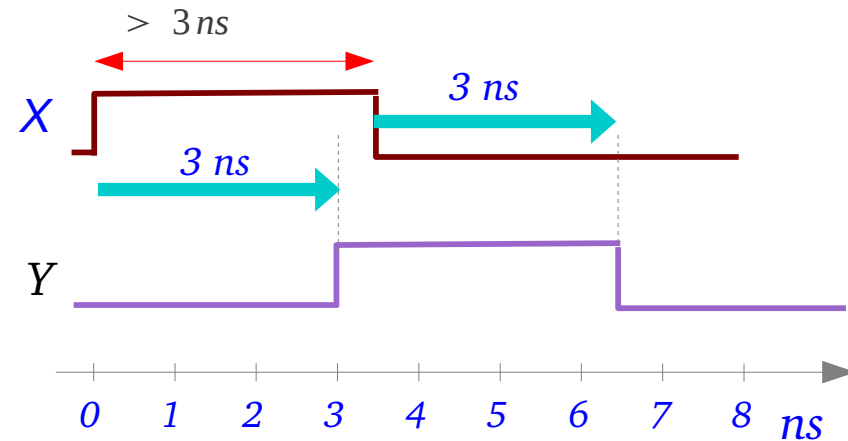
Multiple Concurrent Assignments

```
architecture arch of entity ent is
    • • •
begin

    X1  <=        A or B ;
    X1  <=        C or D ;

        process (A, B, C, D, E, F)
            variable Z2 : bit;
        begin

            Y2  <=        A or B ;
            Y2  <=        C or D ;


            Z2  :=        A or B ;
            Z2  :=        C or D ;


        end process;

end
```

Multiple Concurrent Assignments

Multiple Concurrent Assignment is legal only when a resolution function is defined.

(wire-and, wire-or)

Multiple Sequential Assignments

- *Overwrite*
- *Append*
- *Keep*

Multiple Variable Assignments

Variable Z2 has the result of the latest assignments (The new assignment overwrites the old one)

Resolution Function

```vhdl
architecture arch of entity ent is
    FUNCTION w_and (drivers : bit_vector) RETURN bit is
    BEGIN
        • • •
    END w_and;
    SIGNAL X1 : w_and bit;
        • • •
begin

    X1  <=        A or B ;
    X1  <=        C or D ;


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

        • • •

    end process;

end
```

Multiple Concurrent Assignment is *legal* *only when a* *resolution* *function* *is defined.*

(*wire-and, wire-or*)

```
process (A, B)
begin

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

X1 → w_and

```
process (C, D)
begin

X1  <=   C or D ;
end process;
```
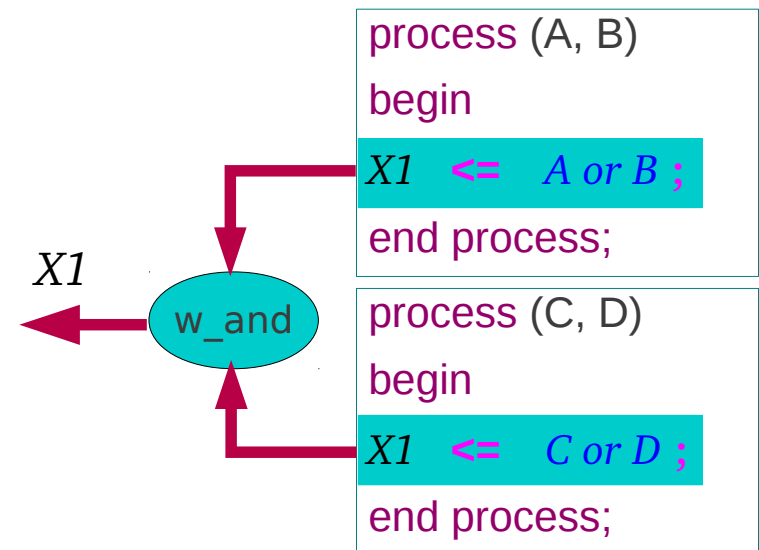
$X1$ <= w_and (*A or B, C or D*) ;

Inertial Delay

Multiple Concurrent Assignments

X2 **<=**   *'1'* **after** *5 ns;*   *Wire-or*
X2 **<=**   *'0'* **after** *3 ns;*   *resolution*
                                                function

process (…)

begin

● ● ●
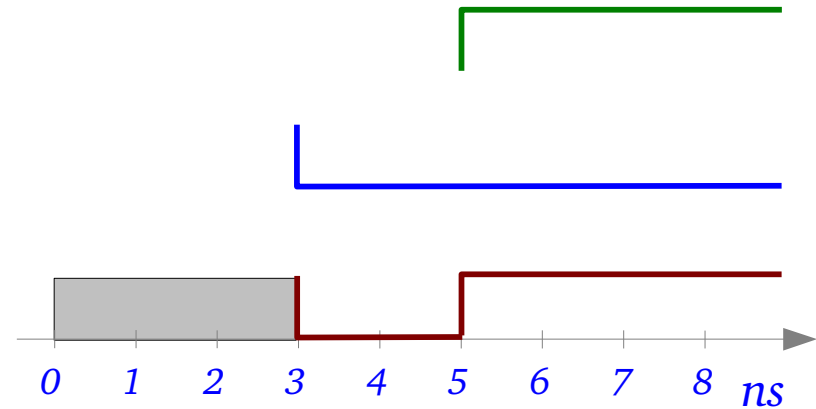
end process;

X2 **<=**   *'1'* **after** *3 ns;*   *Wire-or*
X2 **<=**   *'0'* **after** *5 ns;*   *resolution*
                                                function

process (…)

begin

● ● ●
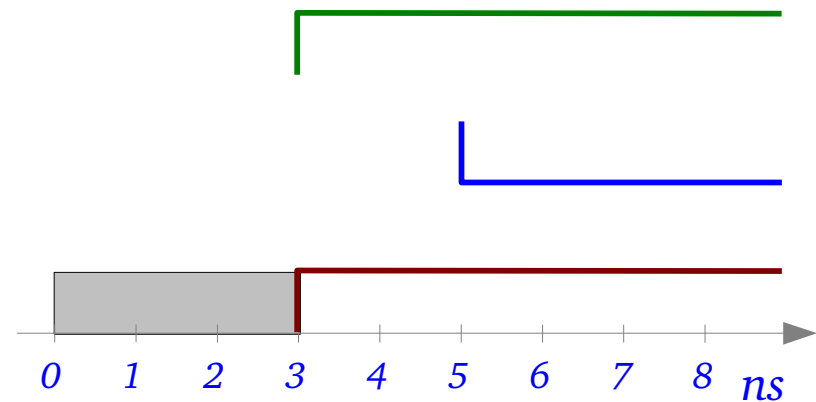
end process;
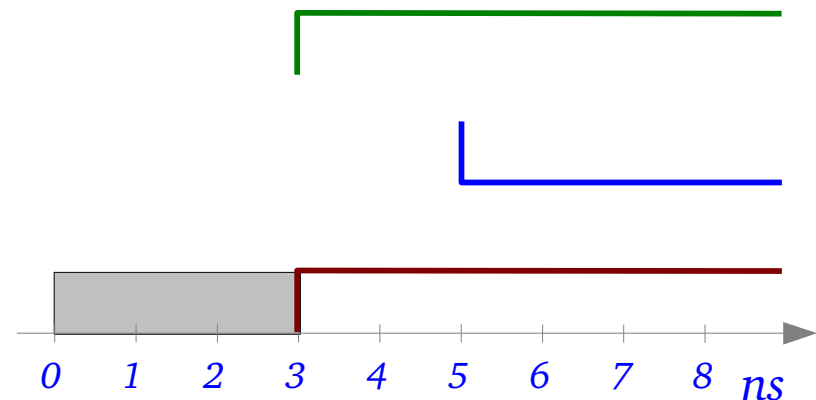
Transport Delay

Multiple Concurrent Assignments

X2 **<= transport** *'1'* **after** *5 ns;*
X2 **<= transport** *'0'* **after** *3 ns;*

process (…)

begin

● ● ●

end process;

Wire-or resolution function

0 1 2 3 4 5 6 7 8 *ns*
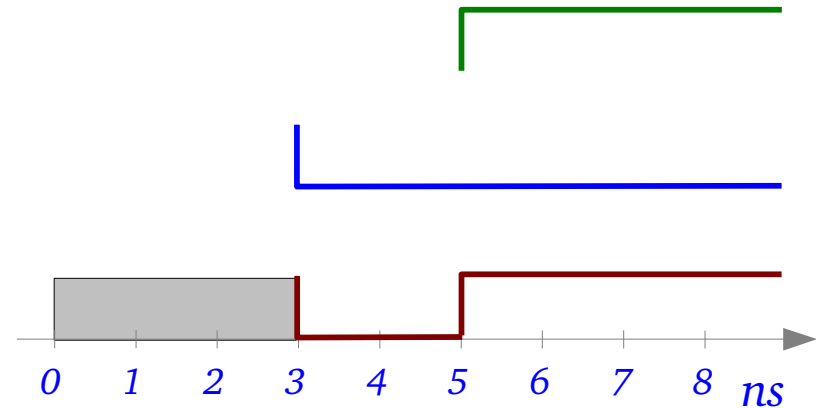
X2 **<= transport** *'1'* **after** *3 ns;*
X2 **<= transport** *'0'* **after** *5 ns;*

process **(…)**

begin

● ● ●

end process;

Wire-or resolution function

0 1 2 3 4 5 6 7 8 *ns*

Inertial Delay

Multiple Concurrent Assignments

X2 **<=** *A* **after** *5 ns;*
X2 **<=** *A* **after** *3 ns;*

Wire-or resolution function

process (…)

begin

● ● ●

end process;

X2 **<=** *A* **after** *3 ns;*
X2 **<=** *A* **after** *5 ns;*

Wire-or resolution function

process (…)

begin

● ● ●

end process;

A

X2

0 1 2 3 4 5 6 7 8 *ns*

Transport Delay

Multiple Concurrent Assignments

X2   **<=**   *A*   **after** *5 ns;*   *Wire-or*
X2   **<=**   *B*   **after** *3 ns;*   *resolution*
   *function*

process (…)

begin

● ● ●

end process;

X2   **<=**   *A*   **after** *3 ns;*   *Wire-or*
X2   **<=**   *B*   **after** *5 ns;*   *resolution*
                                                 function

process (…)

begin

● ● ●

end process;

Signals & Variable Assignment Example 1
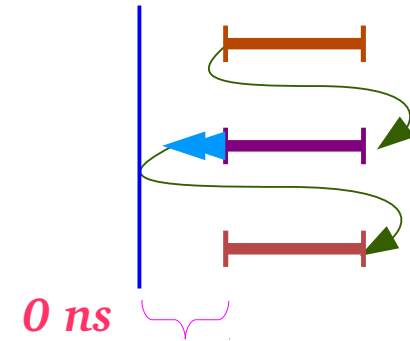
process (A, B, C)

variable *V1, V2* **: bit;**

begin

V1 **:=** *A or B* ;

Y1 **<=** *V2* ;

V2 **:=** *V1 or C* ,

end process;

Δ time *Evaluation*

0 ns

Y1 **<=** *A or B* ;

---

process (A, B, C)

variable *V1, V2* **: bit;**

begin

V1 **:=** *A or B* ;

V2 **:=** *V1 or C* ,

Y1 **<=** *V2* ;

end process;

Δ time *Evaluation*

0 ns
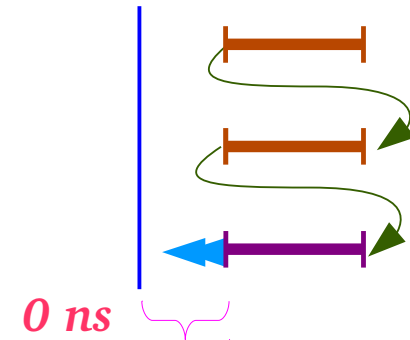
Y1 **<=** *A or B or C* ;

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/