Signals & Variables (2A)

Inertial & Transport Delay Models

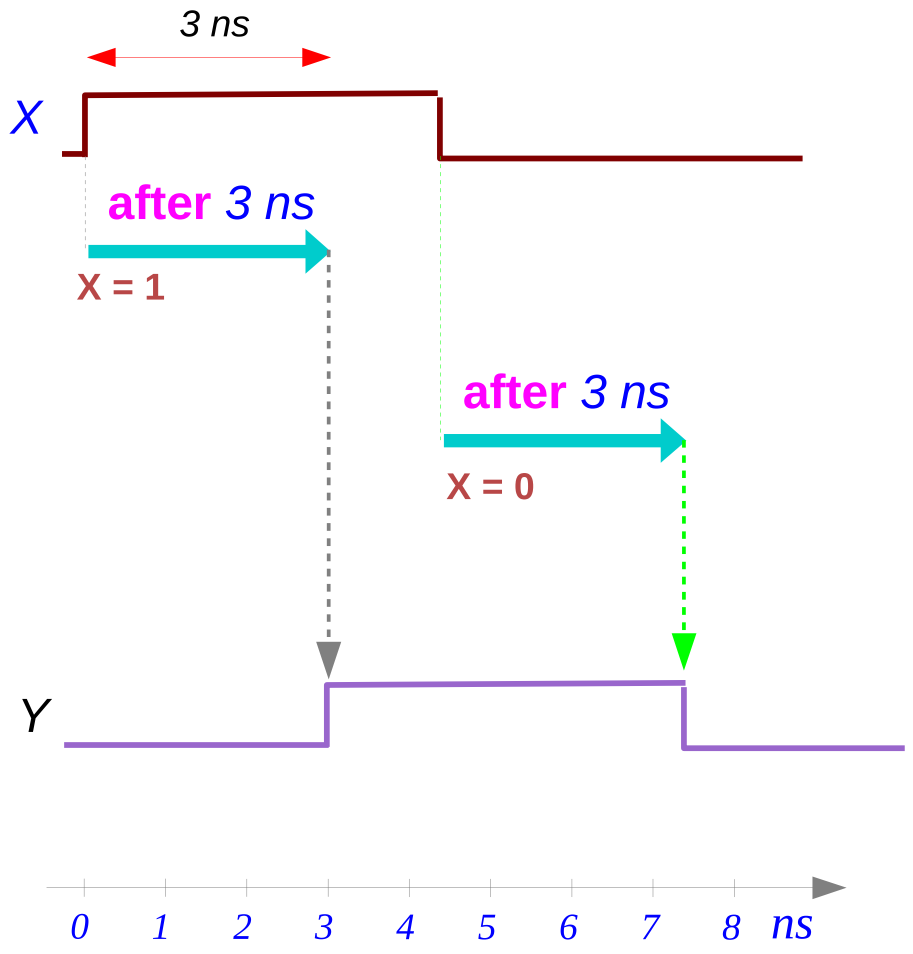
Young Won Lim
07/04/2012

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

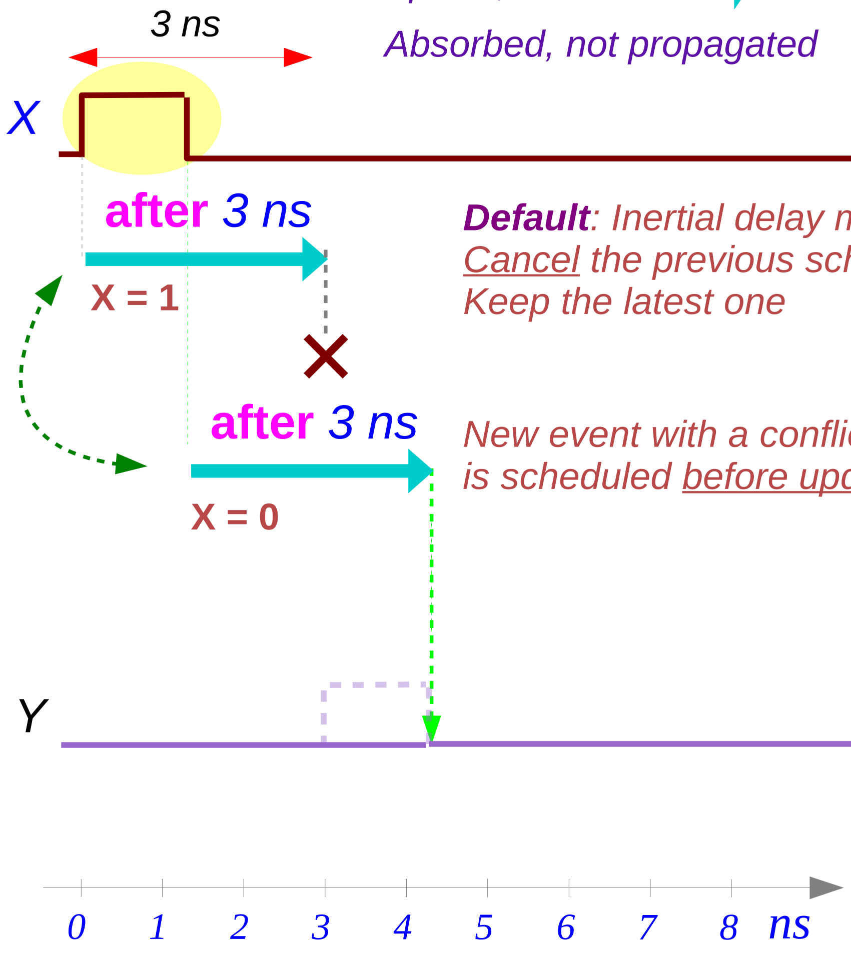
This document was produced by using OpenOffice and Octave.

Inertial Delay

```
Y  <=  X  after 3 ns;
```



Spikes, Glitches → *Absorbed, not propagated*

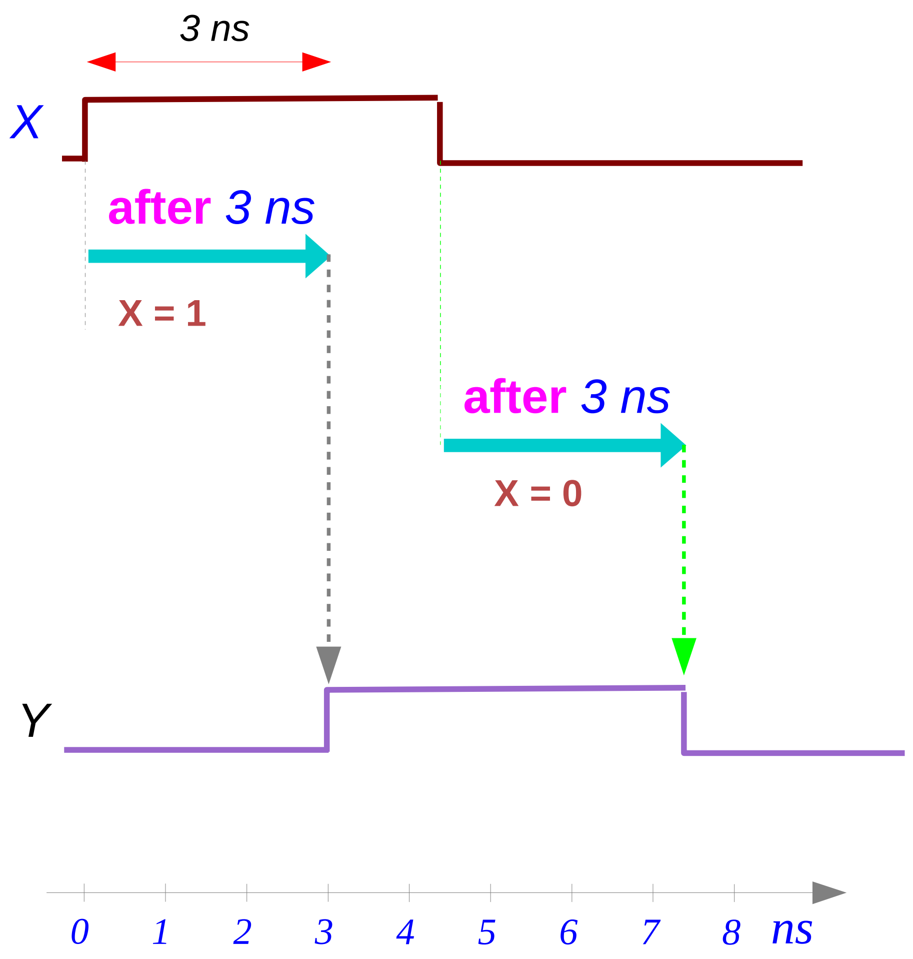


***Default**: Inertial delay mode*
<u>Cancel</u> the previous schedules
Keep the latest one

New event with a conflict value
is scheduled <u>before updating</u>

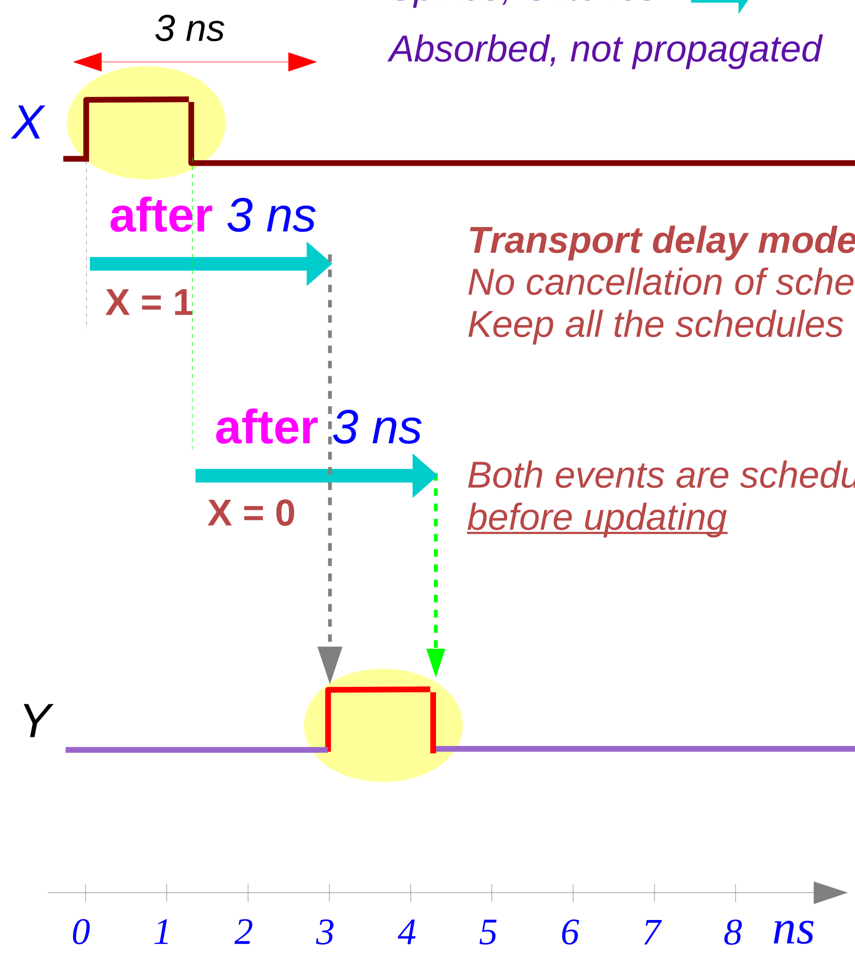
Transport Delay

```
Y  <=  transport X  after 3 ns;
```



Spikes, Glitches

Absorbed, not propagated



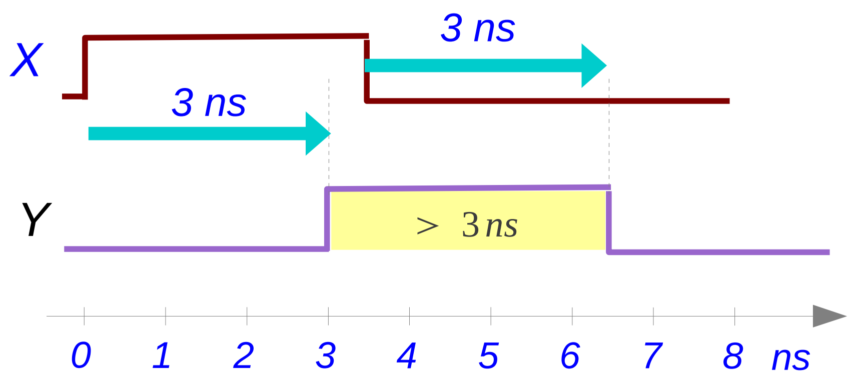
Transport delay mode
No cancellation of schedules
Keep all the schedules

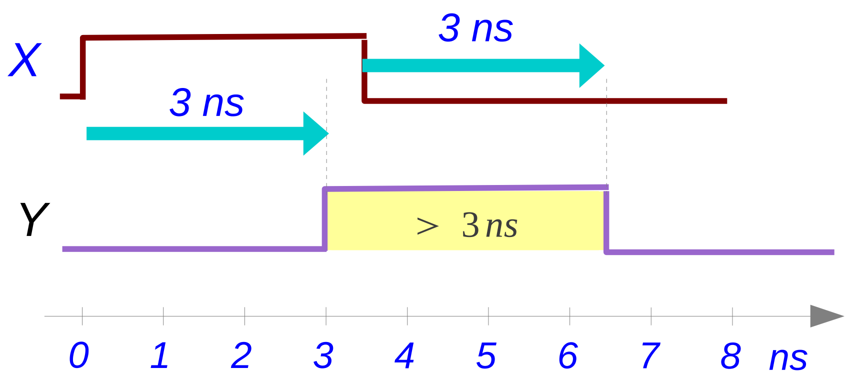
Both events are scheduled
before updating

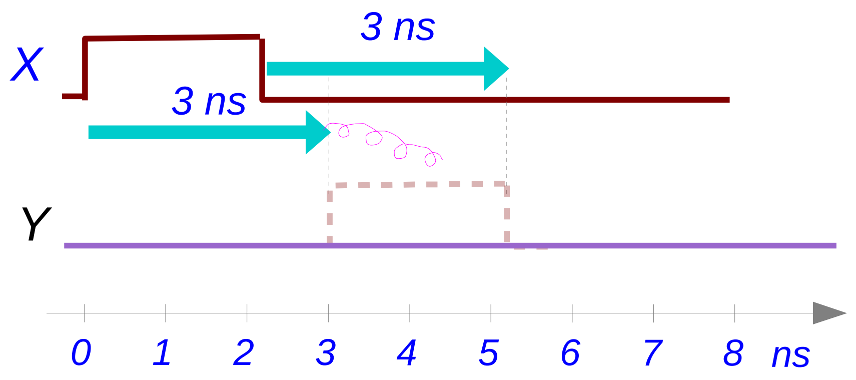
Inertial Delay & Transport Delay

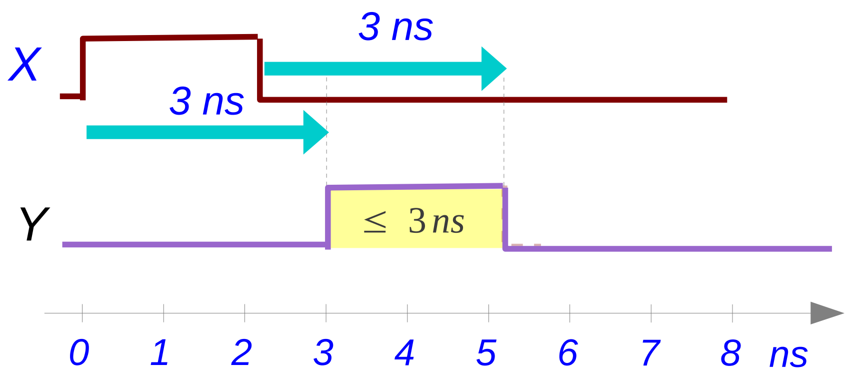
```
Y  <=  X  after 3 ns;
```

```
Y  <=  transport X  after 3 ns;
```









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 & Transport Delay Model (1)

Inertial Delay

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

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

Transport Delay

- *The simulation time of a **new event***
  - ***Before** the time of an **old one***
    - *New one overwrites*
  - ***After** the time of an **old one***
    - *New one is appended*

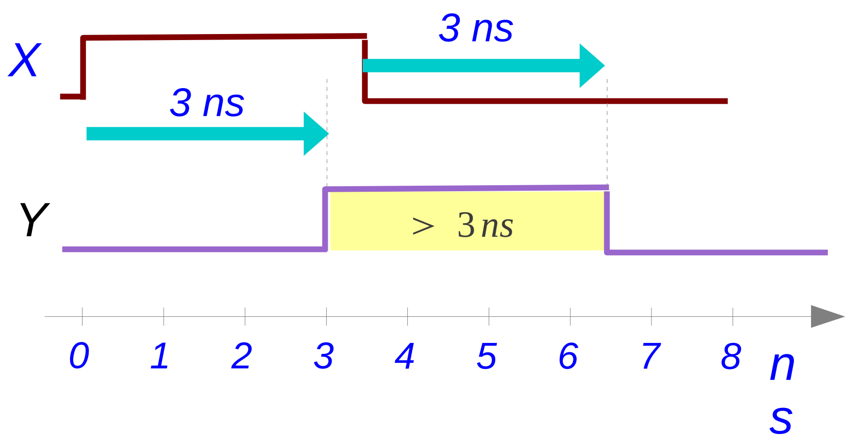
$t2 < t1$	*New one overwrites*
$t1 < t2$	*New one is **appended***

Inertial & Transport Delay Model (2)

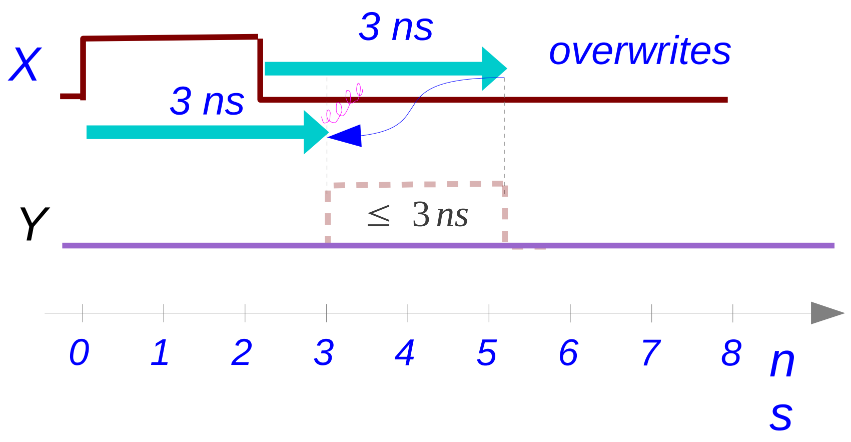
Inertial Delay

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

Y <= X **after** 3 ns;



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



Inertial & Transport Delay Model (3)

Transport Delay

The simulation time of a ***new event***

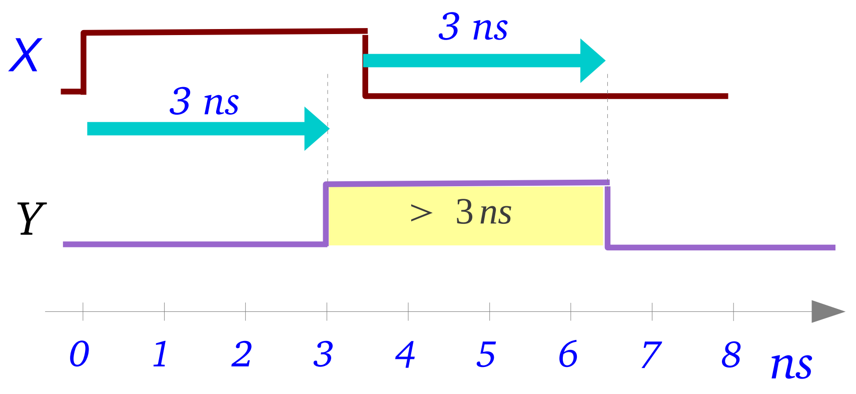
Before *the time of an* ***old one***

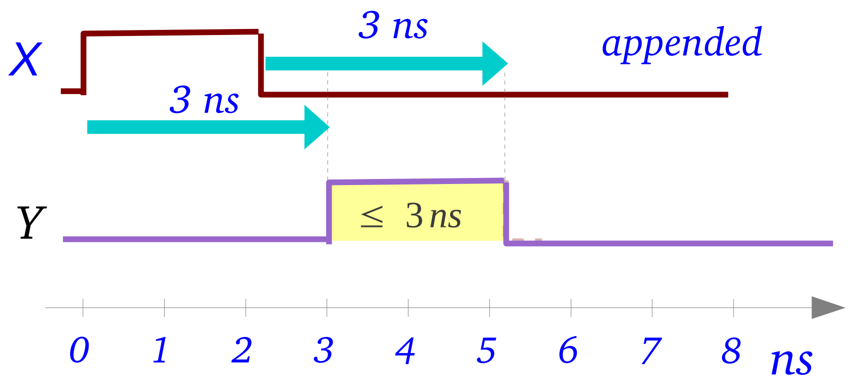
New one overwrites

After *the time of an* ***old one***

New one is appended

`Y <= transport X after 3 ns;`





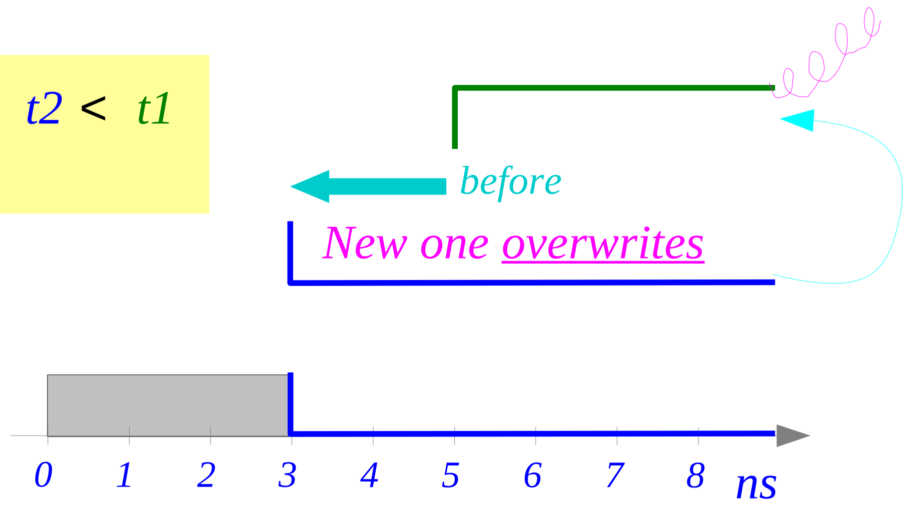
$t2 < t1$	*New one overwrites*
$t1 < t2$	*New one is* ***appended***

Inertial Delay (1)

Multiple Sequential Assignments

```
process (…)
begin

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

end process;
```

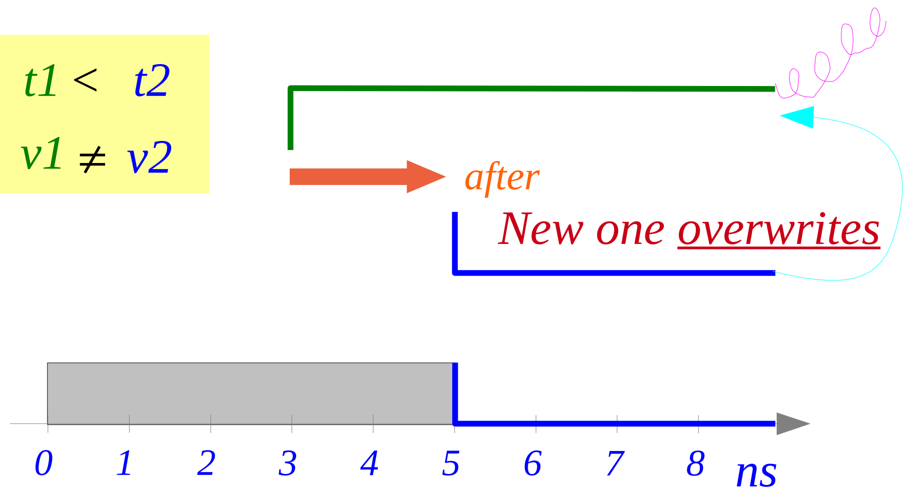
$t2 < t1$



```
process (…)
begin

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

end process;
```

$t1 < t2$

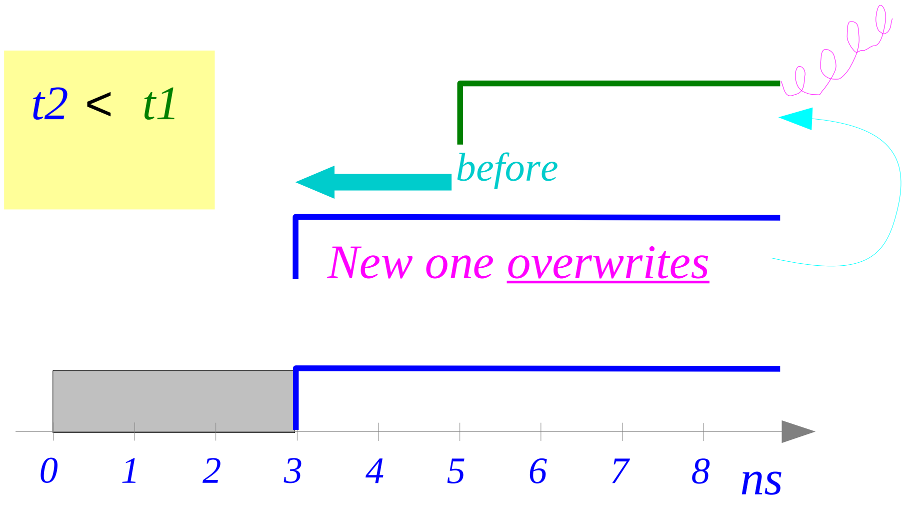
$v1 \neq v2$



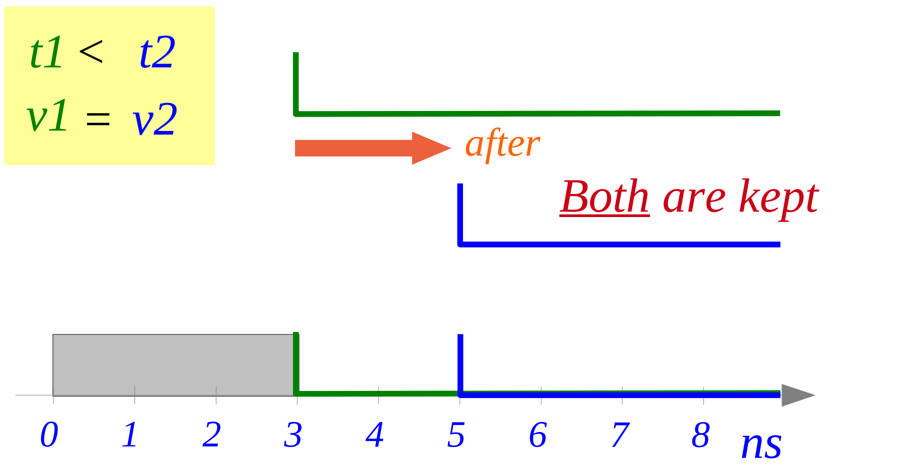
Inertial Delay (2)

Multiple Sequential Assignments

```
process (…)
begin

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

end process;
```



```
process (…)
begin

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

end process;
```

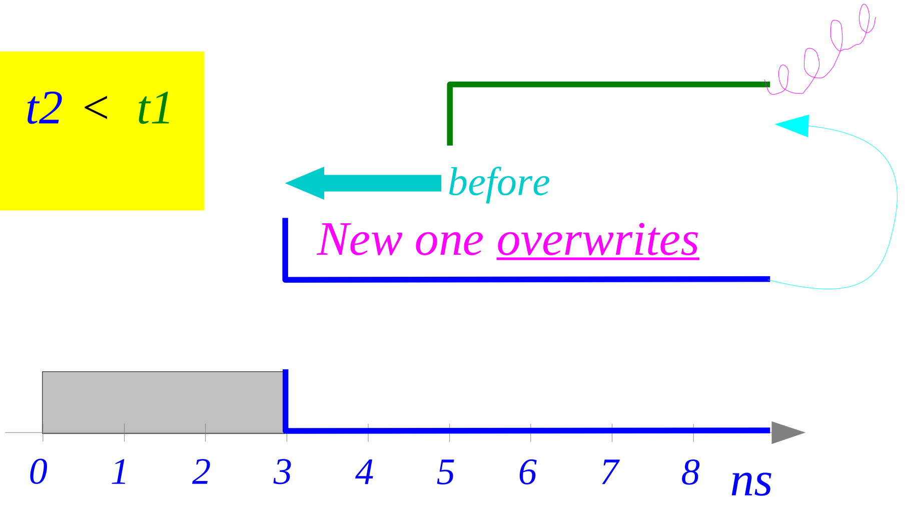


Transport Delay (1)

Multiple Sequential Assignments

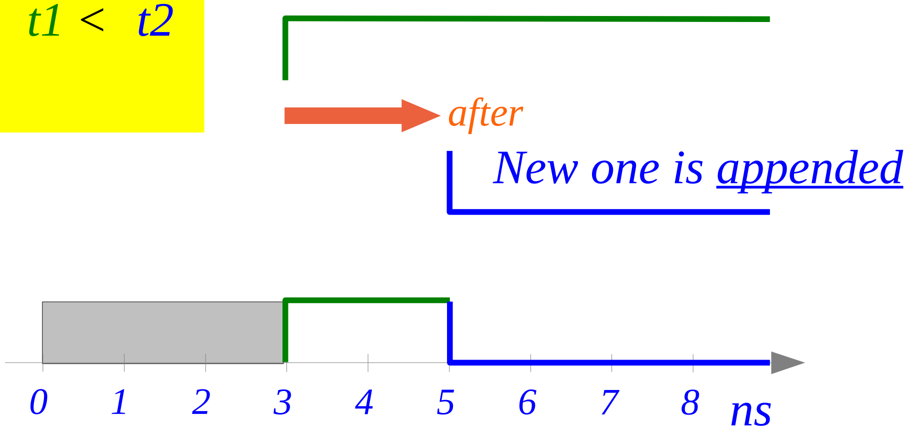
```
process (…)
begin

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

end process;
```

$t2 < t1$



```
process (…)
begin

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

end process;
```

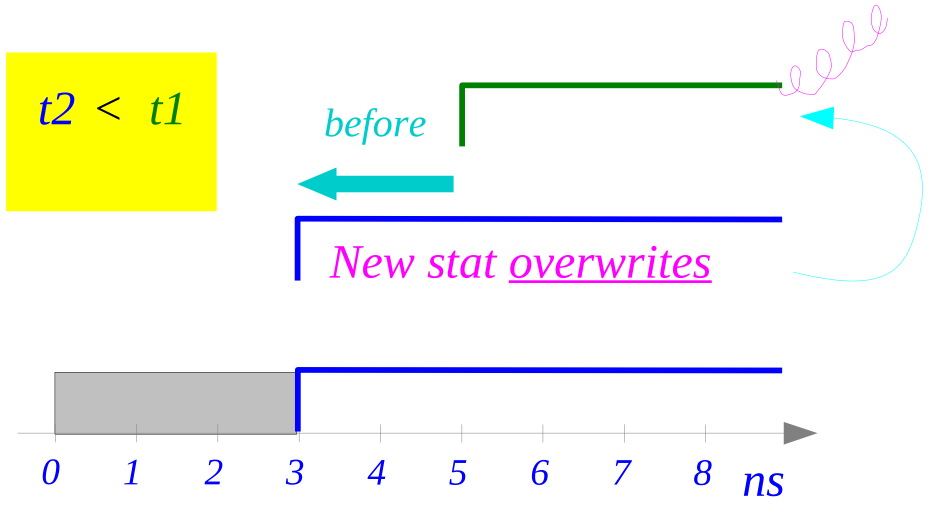
$t1 < t2$



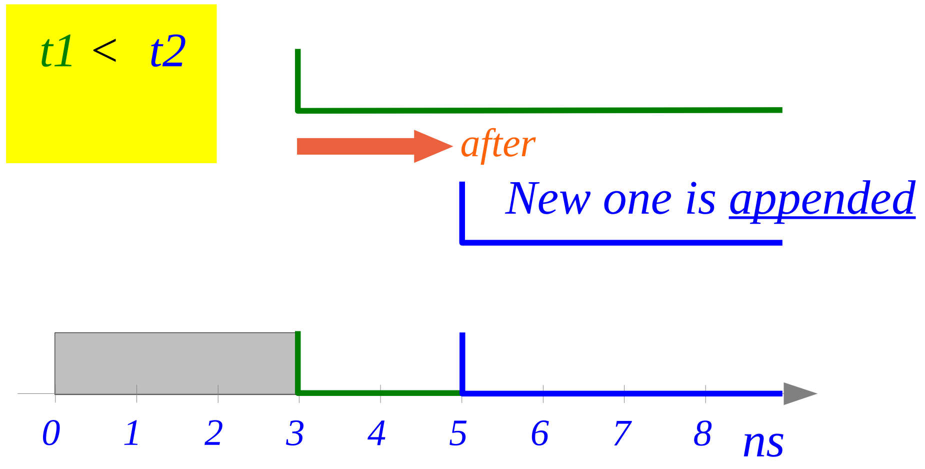
Transport Delay (2)

Multiple Sequential Assignments

```
process (...)
begin

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

end process;
```



```
process (...)
begin

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

end process;
```



Inertial Delay

Multiple Sequential Assignments – Inertial Delay

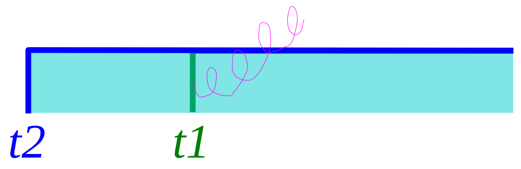
```
process (…)
begin

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

end process;
```

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

before

$t2 < t1$

$v1 = v2$

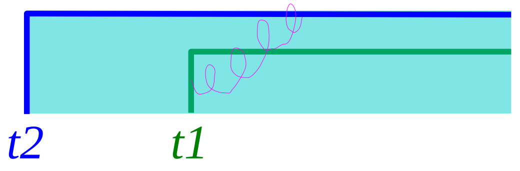


New one overwrites

before

$t2 < t1$

$v1 \neq v2$

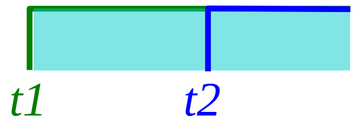


New one overwrites

after

$t1 < t2$

$v1 = v2$

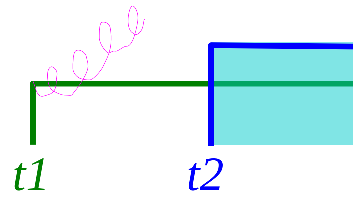


Both are kept

after

$t1 < t2$

$v1 \neq v2$



New one overwrites

Transport Delay

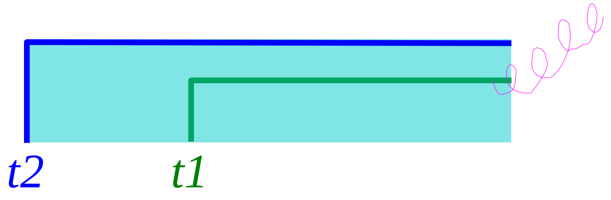
Multiple Sequential Assignments – Transport Delay

```
process (…)
begin

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

end process;
```

$t2 < t1$	*New stat overwrites*
$t1 < t2$	*New stat is appended*

before

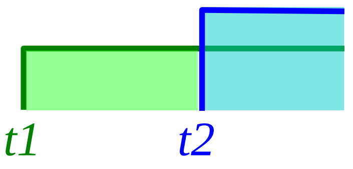
$t2 < t1$



New one overwrites

after

$t1 < t2$



New one is appended

Initial Value

Multiple Concurrent Assignments – Transport Delay

```
architecture arch of entity ent is
signal test : STD_LOGIC := '0';
begin
    test <= transport '1' after 3 ns;
    test <= transport '0' after 5 ns;
end arch;
```

Default Value:

'0' is a default value for any driver

```
test <= transport '0', '1' after 3 ns;
test <= transport '0', '0' after 5 ns;
```

After 3 ns, there are actually two active drivers;
One which drives '0'
The other which drives '1'.

0 at 0 ns
X at 3 ns
X at 5 ns

```
architecture arch of entity ent is
signal test : STD_LOGIC := 'Z';
begin
    test <= transport '1' after 3 ns;
    test <= transport '0' after 5 ns;
end arch;
```

```
test <= transport 'Z', '1' after 3 ns;
test <= transport 'Z', '0' after 5 ns;
```

Z at 0 ns
1 at 3 ns
X at 5 ns

Std_logic Resolution Function Table

	U	X	0	1	Z	W	L	H	D
U	U	U	U	U	U	U	U	U	U
X	U	X	X	X	X	X	X	X	X
0	U	X	0	X	0	0	0	0	X
1	U	X	X	1	1	1	1	1	X
Z	U	X	0	1	Z	W	L	H	X
W	U	X	0	1	W	W	W	W	X
L	U	X	0	1	L	W	L	W	X
H	U	X	0	1	H	W	W	H	X
D	U	X	X	X	X	X	X	X	X

(leftmost literal,
default initial value)

'U'	uninitialized
'X'	forcing unknown
'0'	forcing 0
'1'	forcing 1
'Z'	high impedance
'W'	weak unknown
'L'	weak 0 (pulldown)
'H'	weak 1 (pullup)
'-'	don't care

(used for synthesis only)

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.

Multiple Sequential Assignments

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

Multiple Variable Assignments

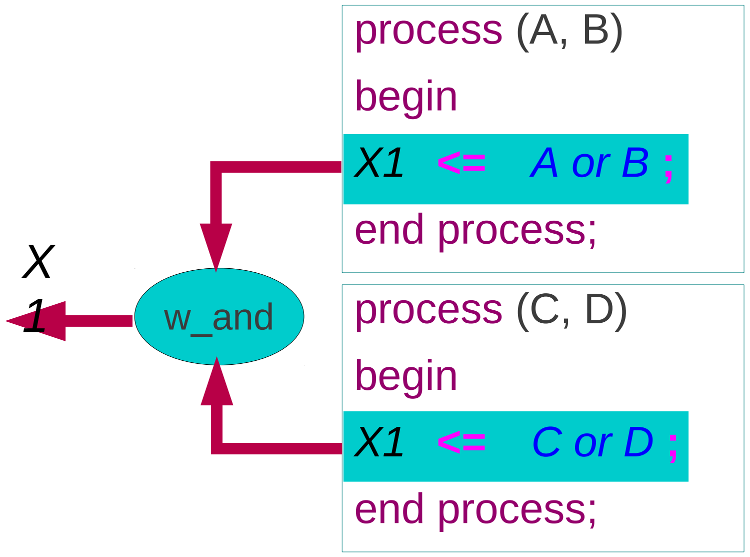
Variable Z2 has the result of the latest assignments (The new assignment overwrites the old one)

Resolution Function

```
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 *<u>legal</u>* *only when a* *<u>resolution function</u>* *is defined.*

(wire-and, wire-or)



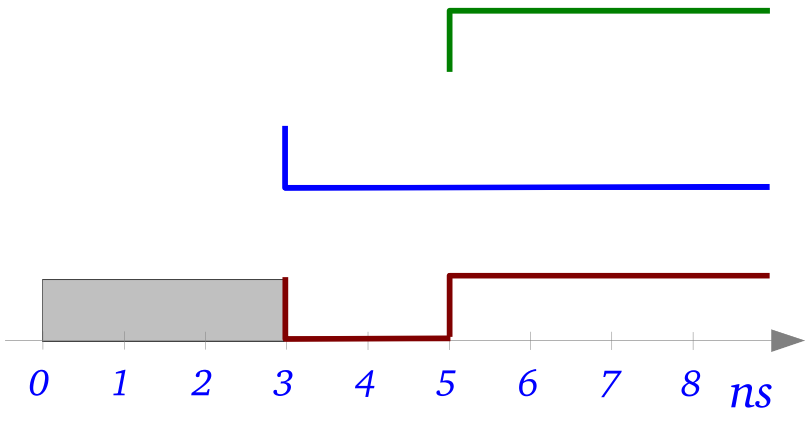
X1 <= w_and (A or B, C or D) ;

Inertial Delay

Multiple Concurrent Assignments

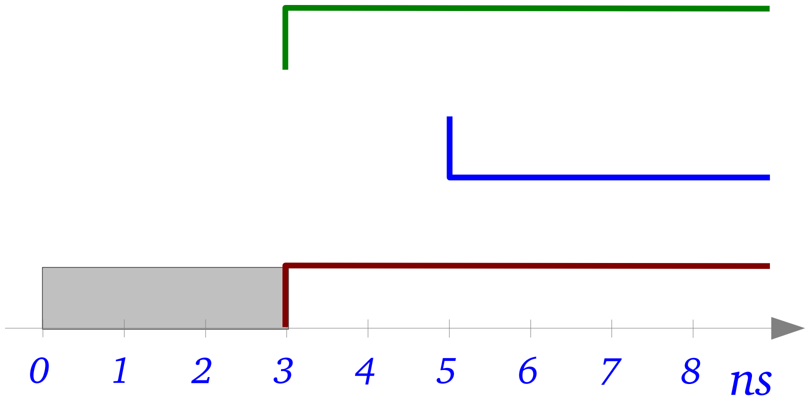
```
X2  <=     '1'  after 5 ns;
X2  <=     '0'  after 3 ns;

process (…)
begin
     • • •
end process;
```

Wire-or resolution function



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

process (…)
begin
     • • •
end process;
```

Wire-or resolution function

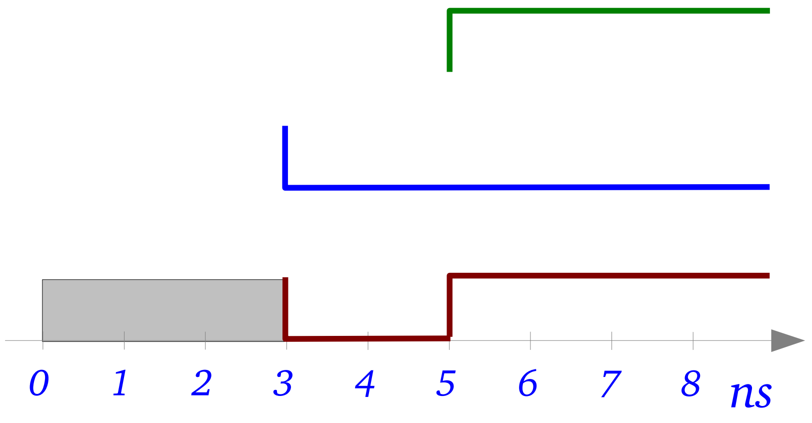


Transport Delay

Multiple Concurrent Assignments

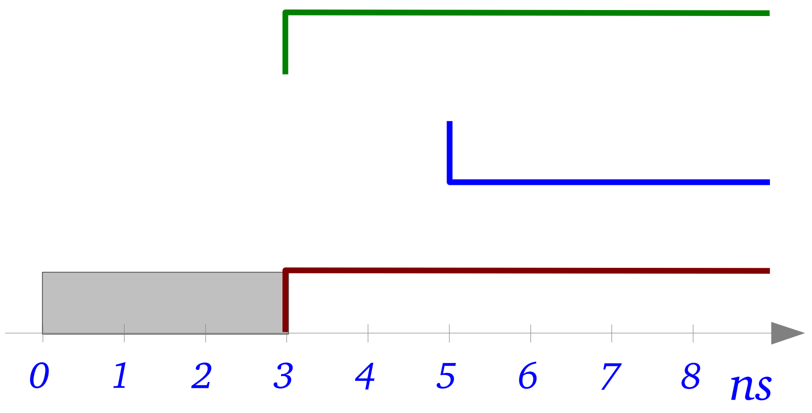
```
X2  <= transport  '1'  after 5 ns;
X2  <= transport  '0'  after 3 ns;
process (…)
begin
    • • •
end process;
```

Wire-or resolution function



```
X2  <= transport  '1'  after 3 ns;
X2  <= transport  '0'  after 5 ns;
process (…)
begin
    • • •
end process;
```

Wire-or resolution function

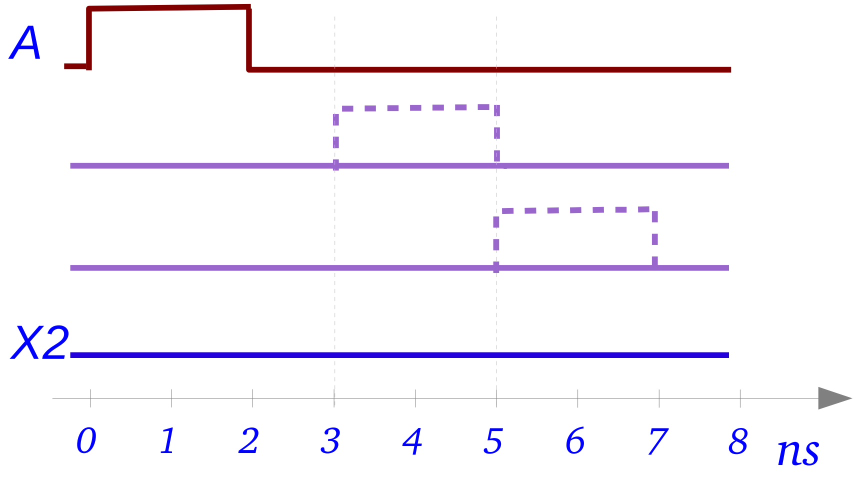


Inertial Delay

Multiple Concurrent Assignments

```
X2  <=   A  after 5 ns;
X2  <=   A  after 3 ns;
process (…)
begin
    • • •
end process;
```

Wire-or resolution function



```
X2  <=   A  after 3 ns;
X2  <=   A  after 5 ns;
process (…)
begin
    • • •
end process;
```

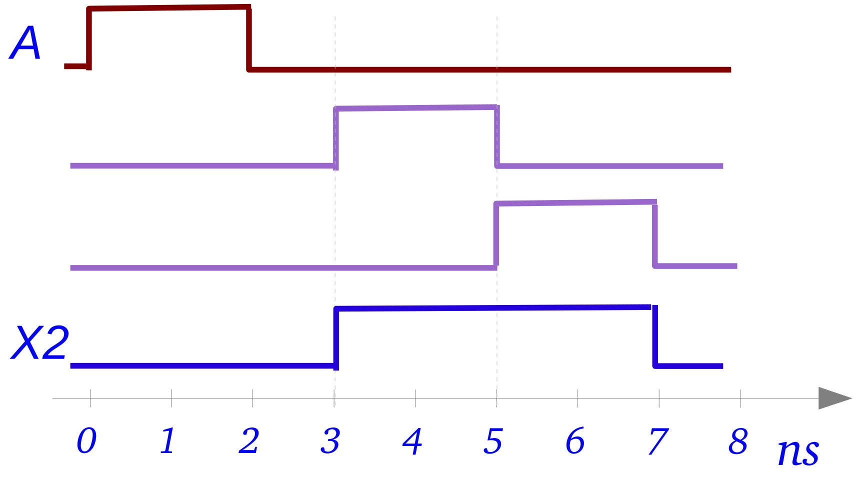
Wire-or resolution function

Transport Delay

Multiple Concurrent Assignments

```
X2  <=   A  after 5 ns;
X2  <=   B  after 3 ns;
process (…)
begin
    • • •
end process;
```

Wire-or resolution function



```
X2  <=   A  after 3 ns;
X2  <=   B  after 5 ns;
process (…)
begin
    • • •
end process;
```

Wire-or resolution function

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/
[8] compgroups.net