bus

waveforms

transport

delta and simulation

Time Modelling and Data Flow Descriptions

Modeling time in VHDL
- Different models of time delay

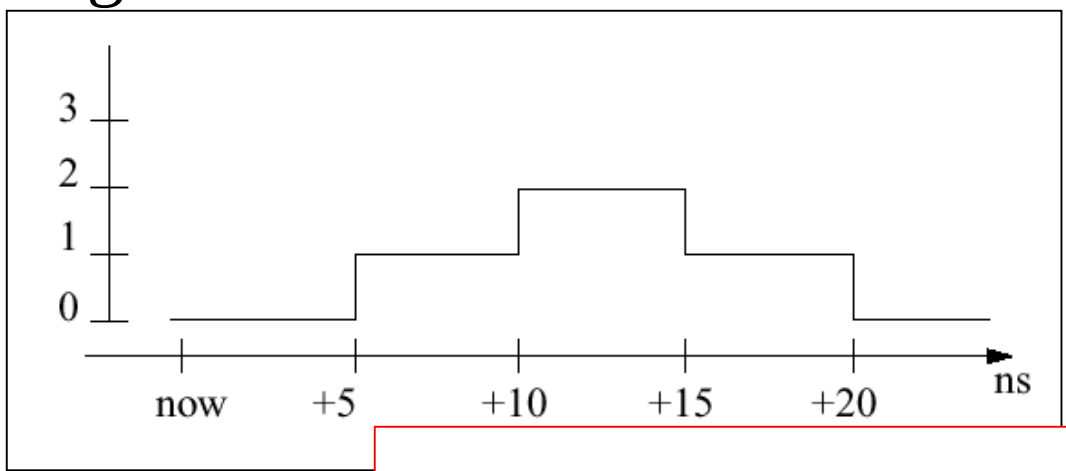
- Specify timing requirement
  - Data flow descriptions
  - Signal resolution
  - Guarded signals

Modelling Timing in VHDL

- VHDL can be used to specify different aspects of timing characteristics of hardware devices:
  - propagation delay of signals
  - operational time
- **Why we need timing?**
  - The type "time" is a pre-defined physical type.
  - Mainly useful for modeling device timing characteristics
  - Can also be used to specify timing requirements, e.g., setup and hold times of devices.
  - You can parameterize timing properties of an entity.

Waveform and Driver

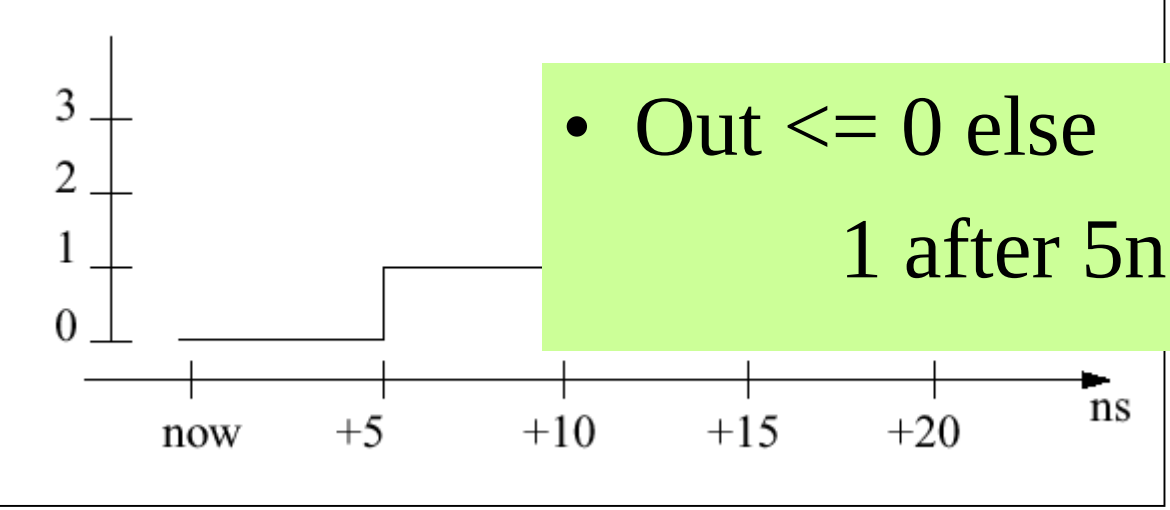
- **Simulator uses drivers for signals**
- **A** driver of a signal contains a current value and a waveform representing projected future values.
- Waveform elements are appended to a driver whenever a signal assignment is executed.



How to describe a waveform

step ←

0	1	2	1	0
	+5 ns	+10 ns	+15 ns	+20 ns



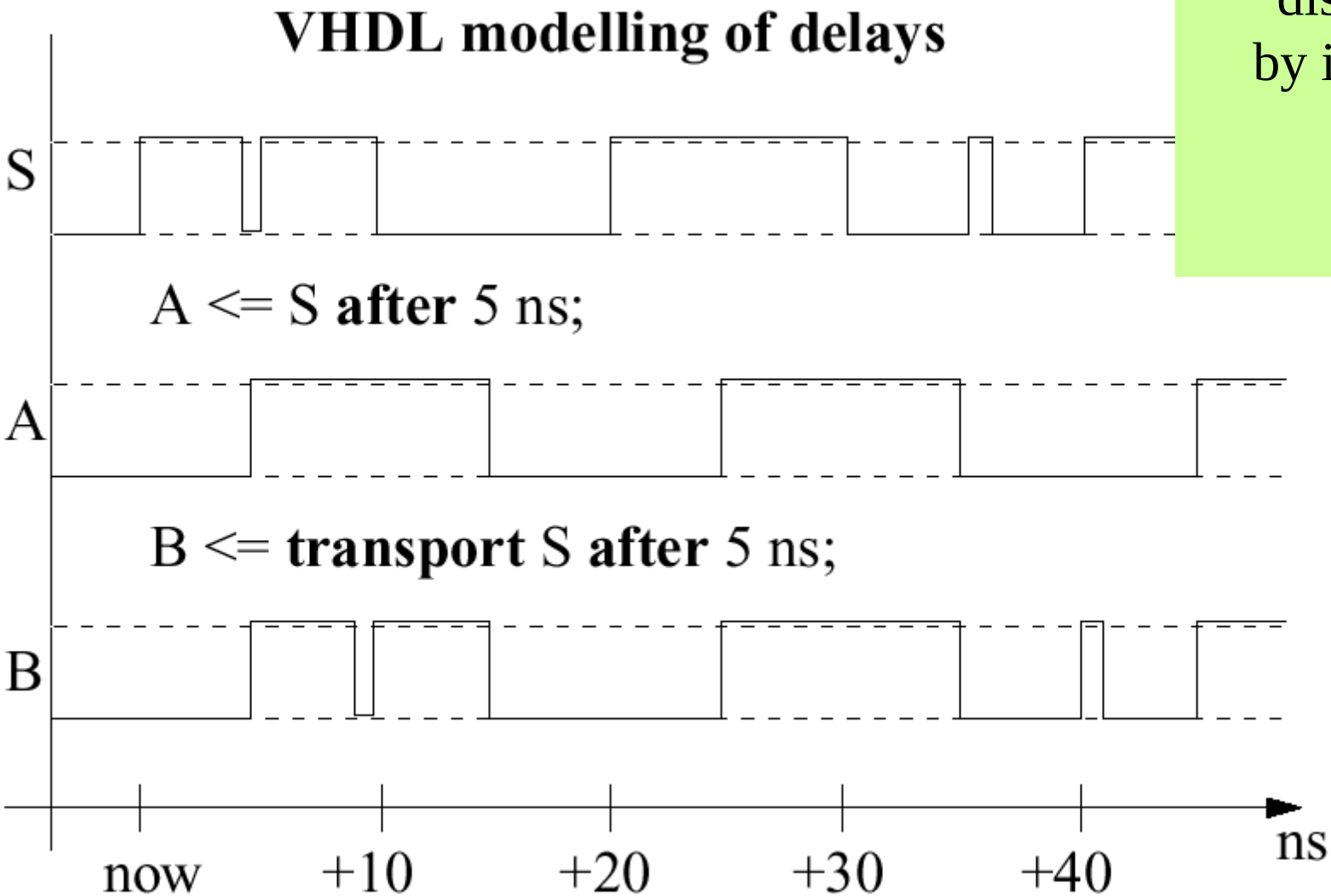
- Out <= 0 else
  1 after 5ns

Models of Time Delay

- Inertial delay
  - Model the time lag between stable inputs and valid output of a device
  - Representative of combinational logic elements
  - Pulses smaller than transmission delay are suppressed
  - Default model for VHDL descriptions
- Transport delay
  - Model a pure delay mechanism
  - All pulses are transmitted
  - Used for transmission lines or elements with clock- cycle latency

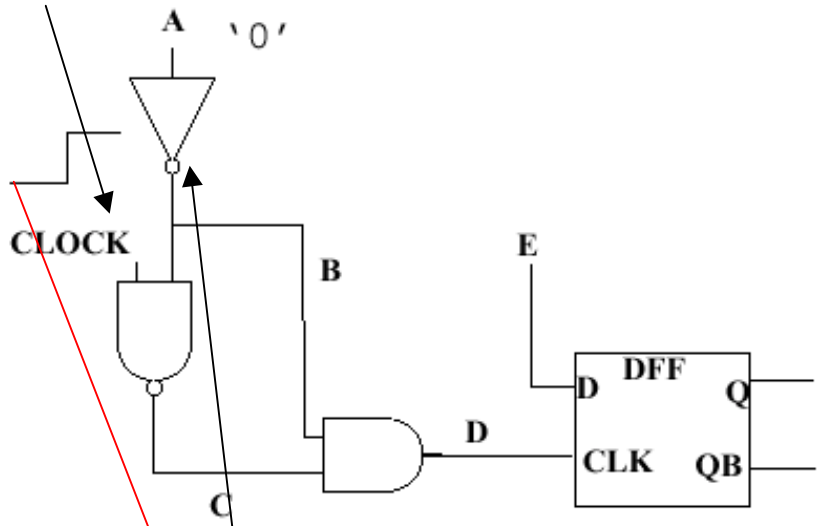
Inertial versus transport delay

How small should be the glitch to be distinguished by inertial and transport?



Time modelling- delta delay .

1. What is wrong with old simulators?



- With this order of evaluation a glitch in D is created

As we see, timing behavior simulated depends on the gate evaluation order

AND first evaluation

evaluate inverter
B<=1
evaluate AND (C=1)
D<=1
evaluate NAND
C<=0
evaluate AND
D<=0

again

'0'

NAND first evaluation

evaluate inverter
B<=1
evaluate NAND
C<=0
evaluate AND
D<=0

'0'

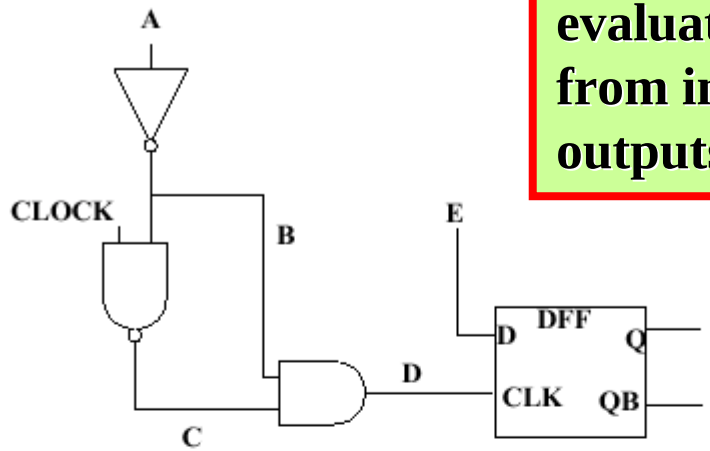
Time modelling- delta delay .

Delta delay of VHDL solves the problem.

- Many delta units of time passed but only one unit of time reported to the user

Delta is as close to zero as we want

This is levelized evaluation from inputs to outputs



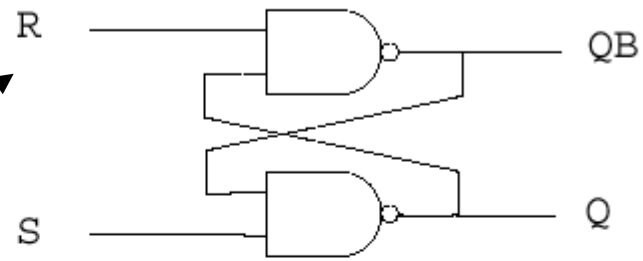
Delta Delay Mechanism

time 10ns	delta(1)	A<=0 evaluate inverter
	delta(2)	B<=1 evaluate AND evaluate NAND
	delta (3)	D<=1 C<=0 evaluate AND
	delta(4)	D<=0
time 11ns		

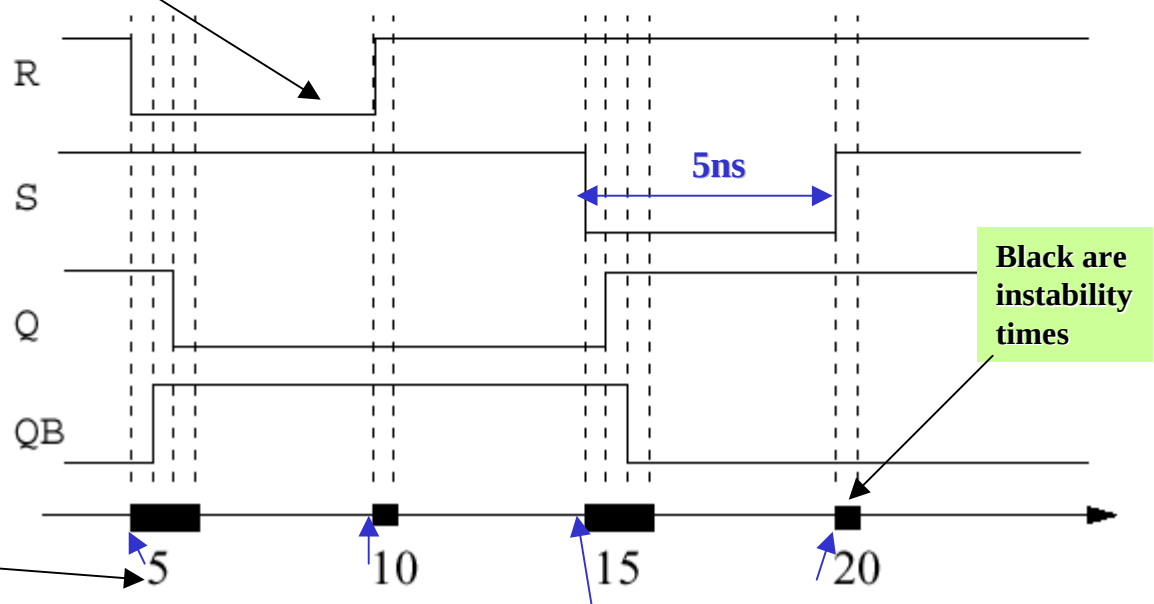
Delta Delay

- If no delay time is specified, a delta delay is assumed for any signal assignment.
- Delta delay represents an infinitesimal delay, less than any measurable time (i.e., femtoseconds), but still larger than zero.
- **An example**

```
QB <= R nand Q;
Q <= S nand QB;
```




```
R <= '0' after 5ns, '1' after 10ns;
S <= '0' after 15ns, '1' after 20ns;
```



Rise/Fall Delay Example

```
architecture rf_delay of half_add is

-- a description of half_add that uses
-- a "look-up-table" for
-- differentiating rise and fall delays

  type table is array (bit) of time;
  constant op_delay: table := (10ns, 7ns);

  begin

    s <= a xor b after op_dealy(s);
    co <= a nand b after op_delay(co);

  end rf_delay;
```

wrong

Defining delays

sum

Carry out

Delay of co

Assertion Statements

assert *condition*
report *message*
severity *level*;

- Ex.

assert not (S = '1' and R = '1')
report “S and R are equal to '1'”
severity Error;

- An assertion statement specifies a boolean condition to check, an error message and a severity indication.

Assertion Statements

When the condition is false, **the error message is sent** to the system output with an indication of the severity and the name of the design unit in which the assertion occurred.

- (Default message: “Assertion violation”).

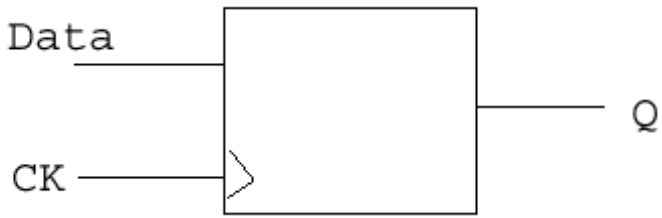
- The severity is of the type Severity_Level which has the values of:
  - Note,
  - Warning,
  - Error,
  - and Failure. (Default se-verity level: Error)
- In some VHDL system, unsatisfied conditions of severity Error or Failure cause the simulation to terminate.

Using Assertions to Specify Timing Requirements

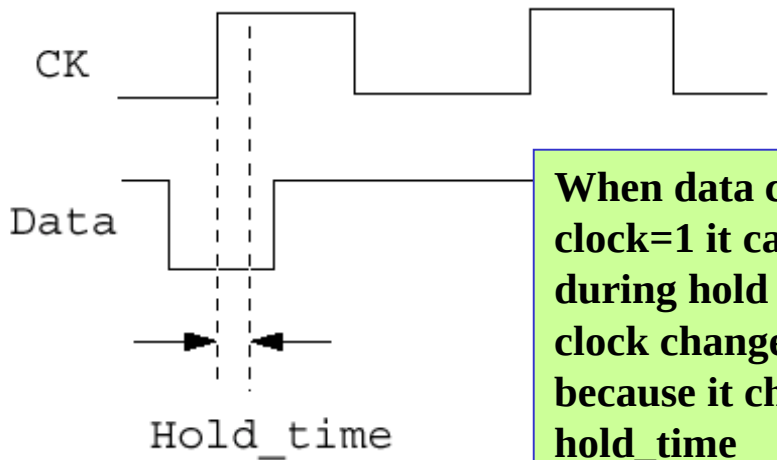
- Assertion statements can be used to specify timing requirements, such as set-up time and hold time.
- Example

```
if not Data'STABLE and CK = '1' then
  assert CK'STABLE(Hold_time)
    report "Data changed within hold time"
    severity Warning;
end if;
```

If this condition is false report is printed



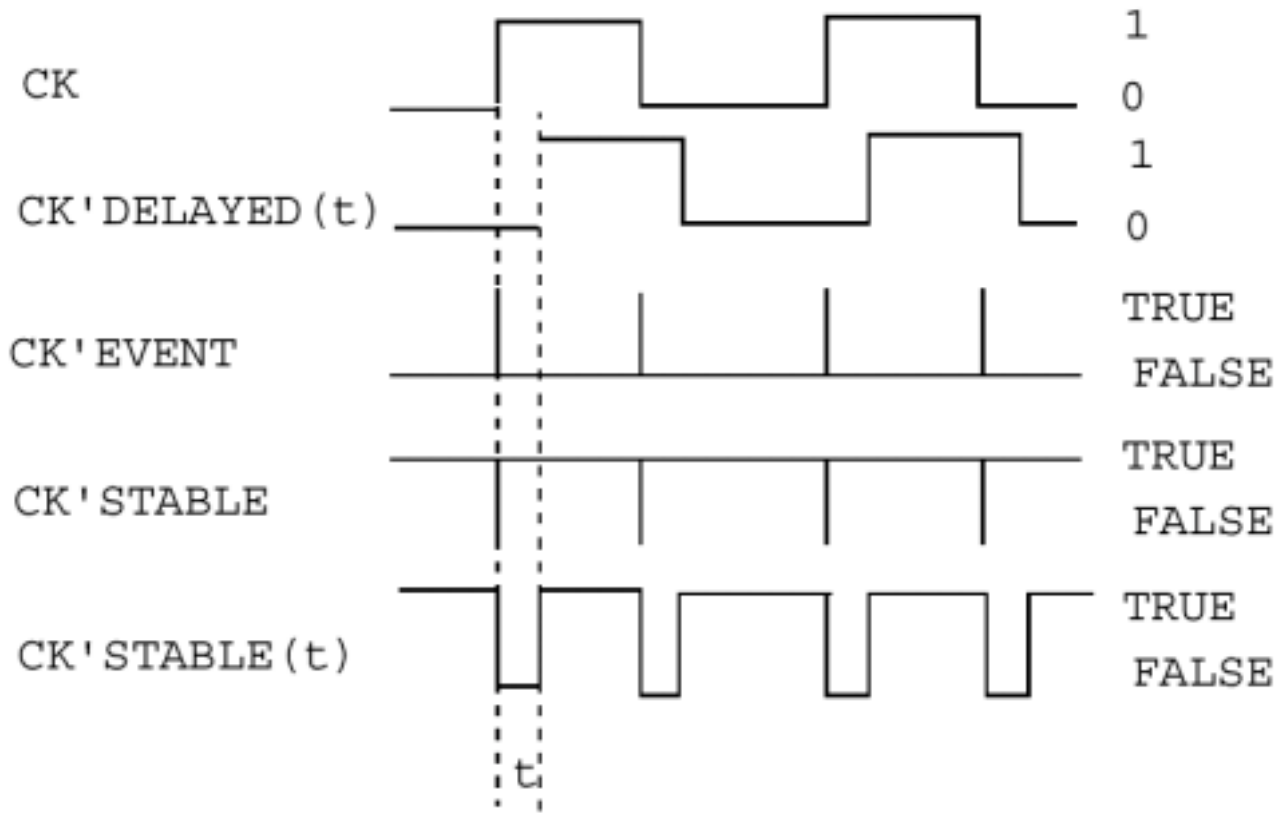
If data is NOT stable and clock=1 then check if clock is stable in hold time



When data changes during clock=1 it cannot change during hold time from clock change, this is OK because it changed after hold_time

Signal -Related Attributes

- VHDL contains a number of predefined attributes which are related to signals.
- They can be divided into two classes:
  - attributes which define signals themselves
  - attributes which are functions to provide information about signals.



These attributes are signals themselves

Data Flow Descriptions in VHDL

- A data flow description consists of a set of concurrent signal assignment statements.
- A signal assignment statement executes in response to change on its input signals.

Data Flow Descriptions in VHDL

Each value in the waveform will be scheduled to appear on the target after the specified delay.

- **If the assignment statement executes again, previously scheduled values may be overridden.**
- **A delay of zero represents an infinitesimally small delay - signal assignment never takes effect immediately.**
- **Data flow descriptions are similar to register-transfer level expressions.**
- **They may imply hardware implementation structure.**

An example:

```
architecture dataflow of full_adder is

begin

  s <= a xor b xor c after 10ns;
  co <= (a and b) or (a and c) or (b and c)
     after 10ns;

end dataflow;
```

Concurrent Signal Assignment

```
target <= transform ;
```

Conditional waveforms

```
target <=   options
            waveform1 when condition1 else
              :
              :
            waveformN-1 when conditionN-1 else
            waveform N ;

options -> [guarded] [transport]
```

Guarded will be discussed in a separate lecture

An Example

```
Y <= transport
          '1' after Delay when A='1' and B='1' else
          '0' after Delay;
```

Concurrent Signal Assignment

Selected Signal Assignment

```
with expression select
target <= options
    waveform1 when choices1,
        :
        :
    waveformN when choicesN;
```

An Example

```
with Sel select
DOut <=
          "00000001" after Delay when "000",
          "00000010" after Delay when "001",
          "00000100" after Delay when "010",
          "00001000" after Delay when "011",
          "00010000" after Delay when "100",
          "00100000" after Delay when "101",
          "01000000" after Delay when "110",
          "10000000" after Delay when "111";
```

- This describes a decoder, or translator from binary to one-hot code

Conditional Signal Assignment

- The waveform of the assignment to a signal can be chosen based on a set of boolean conditions.

```
Count <=
        3 when A = '1' and B = '1' else
        2 after 5ns when A = '1' else
        1 after 5ns when B = '1' else
        0;
```

- When any input changes, the conditions are evaluated again in order.
- The waveform associated with the first true condition is assigned to the signal.
- The last waveform must not have a condition- it is the default.

Selected Signal Assignment

concatenation

- Similar to a case statement.

```
with P1 & P2 select
Deco<= "1000" after 12ns when "00",
       "0100" after 12ns when "01",
       "0010" after 12ns when "10",
       "0001" after 12ns when others;
```

- When any input changes, the expression is evaluated.
- The waveform associated with the value is selected.
- Every possible value must have a waveform.

Sources

- Krzysztof Kuchcinski