Digital systems operate with concurrent signals.

Signals are assigned values at a specific point in time.

VHDL uses **signal assignment** statements.
- Specify value and time
- Multiple **signal assignment** statements are executed concurrently.
VHDL must specify:
- Events
- Delays
- Concurrency

```vhdl
architecture concurrent_behavior of half_adder is
begin
    sum <= (x xor y) after 5 ns;
    carry <= (x and y) after 5 ns;
end concurrent_behavior;
```
CSA Statements

- **concurrent_behavior**
  - Name of architecture that defines the half_adder entity.

- **signal assignment statements**
  - `sum <= (x xor y) after 5 ns;`
  - `carry <= (x and y) after 5 ns;`

  - Signal assignment operator <=
    - Describes how output signals depend on input signals

  - An output signal changes if an input signal has changed.
Half Adder Operation

Event

x

y

sum

carry

5 10 20 25 30 35 40
Signal propagation of the XOR and AND gates must be taken into account.

- Both gates require 5 ns propagation delay.
- Signal assignment statements define this through the `after` Keyword.
- This keyword specifies when the output signal is set to the result of an evaluation after an input signal transition (event).
- The textual order of the assignment statement has no influence on the timing.
library and use clauses

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity 2BA4 is
  ...
end entity 2BA4;
```

- A library contains design entities that be used.
- The `library IEEE` clause defines the IEEE library.
- The library may contain packages.
- The above example specifies through the `use` clause the `IEEE.STD_LOGIC_1164.ALL` packages.
- This package is required for `std_logic` type declaration.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity full_adder is
  Port (in1, in2, c_in:in std_ulogic;
        sum, c_out:out std_ulogic );
end full_adder;

architecture dataflow of full_adder is
signal s1,s2,s3: std_ulogic;
constant gate_delay: Time := 5 ns;

begin
  s1 <= (in1 xor in2) after gate_delay;
  s2 <= (c_in and s1) after gate_delay;
  s3 <= (in1 and in2) after gate_delay;
  sum <= (s1 xor c_in) after gate_delay;
  c_out <= (s2 or s3) after gate_delay;
end dataflow;
Full-Adder Schematic

Full-adder
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity full_adder is
    Port (in1, in2, c_in:in std_ulogic;
        sum, c_out:out std_ulogic );
end full_adder;

architecture dataflow of full_adder is

signal s1, s2, s3: std_ulogic;
constant gate_delay: Time := 5ns;
begin
    s1 <= (in1 xor in2) after gate_delay;
    s2 <= (c_in and s1) after gate_delay;
    s3 <= (in1 and in2) after gate_delay;
    sum <= (s1 xor c_in) after gate_delay;
    c_out <= (s2 or s3) after gate_delay;
end dataflow;
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity full_adder is
  Port (in1, in2, c_in: in std_ulogic;
        sum, c_out: out std_ulogic);
end full_adder;

architecture dataflow of full_adder is
begin
  s1 <= (in1 xor in2) after gate_delay;
  s2 <= (c_in and s1) after gate_delay;
  s3 <= (in1 and in2) after gate_delay;
  sum <= (s1 xor c_in) after gate_delay;
  c_out <= (s2 or s3) after gate_delay;
end dataflow;
```
The Full-Adder Model

- The full-adder simulates the signal transitions at gate-level
- The model has three internal signals
- These signals are not ports to the entity
- The internal signals are declared in the architectural declaration
- The Boolean equations define how each signal is derived as a function of:
  - Other signals
  - Propagation delay
- **Constant** can be used to declare a constant of a particular type.
- In this case **Time**
Signals are not variables

- History of values over time
- Waveform

```vhdl
signal s1: std_ulogic := '0';
```

- Signals may use the assignment symbol := followed by an expression
- The value of the expression will be initial value of the signal
- If no initialisation is provided the signal receives a default value
- VHDL signal types:
  - Integer, real, bit_vector...
A concurrent signal assignment statement (CSA)

\[ \text{sum} \leq (x \text{xor} y) \text{ after } 5 \text{ ns;} \]

In a more general form:

- \[ \text{signal} \leq \text{value expression after time expression;} \]
- In the example,
  - if \( x \) or \( y \) change its value the \text{sum} will be assigned the result of the \((x \text{xor} y)\) evaluation after 5ns.
- The Time-value pair represents the future value of the signal.
- Also called transaction.
Multiple Signal Transactions

It is possible to specify the following:

\[
s1 <= (x \ xor \ y) \text{ after } 5 \text{ ns}, (x \ or \ y) \text{ after } 10 \text{ ns}, (\not{x}) \text{ after } 10 \text{ ns};
\]

- After one of the signals changed all three waveform elements will be evaluated and scheduled according to their after specification.
- The simulation keeps an ordered list of all transactions scheduled for a particular signal.
- The scheduled transactions are also known as: Projected output waveform.
Waveform Specification

The following CSA will generate the following waveform:

\[ s2 \leq '0', '1' \text{ after 10 ns}, '0' \text{ after 20 ns}, '1' \text{ after 40 ns}; \]
Resolved Signals

- In a physical system a wire (signal) has a driver.
- This driver determines the waveform.
- Up to now every signal had one driver only
- But real systems have shared signals:
  - Buses
  - Wired logic
- VHDL determines the value of the signal with multiple drivers through a resolution function.
A shared signal must be declared as a resolved type.
The previous examples used unresolved types:

```
std_ulogic_vector (7 downto 0);
std_ulogic;
```

The following declaration will make these signal types resolved:

```
std_logic_vector (7 downto 0);
std_logic;
```