Synthesis produces a digital circuit that implements the behavior captured in the VHDL description.

VHDL is also the bases for a simulation.

Characteristics of digital systems:
- Structural
- Behavioral
- Physical
Event, Propagation Delays and Concurrency
Signals

- May be 0, 1, or Z
- Equivalent to wires in digital circuits
- May be assigned values
- Signals are associated with time values
- Sequences of values determines the waveform
- Signal type depends on the level of abstraction
- At gate level through wires (or, and, xor...)
- At module level through integer (ALU...)

CS2022, 2nd HDL Lecture, Dr. M. Manzke, Page: 3
Shared Signals

► Hardware description languages must be expressive enough to describe signals that may be driven by one or more sources.

► Bus
Design Entity

Design entities could be:
- Board
- Chip
- Gate
- Transistor

This design component behavior must be:
- Described
- Simulated
Design Entity - Gate Level Example

- **Half-Adder**
  - Input signals: x, y
  - Output signals: sum, carry
Design Entity - Description

- Input signals
- Output signals
- Behavior
  - Truth table
  - Boolean equation
  - Wires between gates
- Two components in the design-entity description:
  - The interface
  - Internal behavior
Entity Declaration

Interface to design entities through

Entity Declaration:

```
entity half_adder is
    Port ( x : in bit;
           y : in bit;
           sum : out bit;
           carry : out bit);
end half_adder;
```

Blue = Keywords

Design Entity Name

Port = input & output
Blue bold type denotes VHDL reserved keywords (entity, port, ...)

VHDL is not case sensitive

Half-adder = HALF-ADDER

Ports define the input and output of the design entity

Ports are signals that enable communication between the design entity and other entities.

Port signals must declare their types.
Port Declaration

Signal types defined in the VHDL language

- **bit**
  - Represents a single-bit signal

- **bit_vector**
  - Represents a vector of signal of type **bit**

- Bit and bit_vector are only two out of several other VHDL data types.
IEEE 1164 standard defines nine-value signals:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Uninitialised</td>
</tr>
<tr>
<td>X</td>
<td>Forcing Unknown</td>
</tr>
<tr>
<td>0</td>
<td>Forcing 0</td>
</tr>
<tr>
<td>1</td>
<td>Forcing 1</td>
</tr>
<tr>
<td>Z</td>
<td>High Impedance</td>
</tr>
<tr>
<td>W</td>
<td>Weak Unknown</td>
</tr>
<tr>
<td>L</td>
<td>Weak 0</td>
</tr>
<tr>
<td>H</td>
<td>Weak 1</td>
</tr>
<tr>
<td>-</td>
<td>Don’t Care</td>
</tr>
</tbody>
</table>
The following modifications are required to make the previous entity declaration IEEE compliant.

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

entity half_adder is
  Port ( x : in std_logic;
         y : in std_logic;
         sum : out std_logic;
         carry : out std_logic);
end half_adder;
```
Signal Mode

Port declaration distinguishes between:

- **in** - input signal
- **out** - output signal
- **inout** - bidirectional signal

```vhdl
entity half_adder is
   Port ( x : in std_logic;
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux is
    Port ( I0 : in std_logic_vector(7 downto 0);
            I1 : in std_logic_vector(7 downto 0);
            I2 : in std_logic_vector(7 downto 0);
            I3 : in std_logic_vector(7 downto 0);
            Sel : in std_logic_vector(1 downto 0);
            Z : out std_logic_vector(7 downto 0));
end mux;
std_logic_vector(7 downto 0)

entity mux is
  Port ( I0 : in std_logic_vector(7 downto 0);

This example refers to 8 bits long input vector.

- bit 7 - most significant bit
- bit 0 - least significant bit
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity half_adder is
  Port ( x : in std_logic;
         y : in std_logic;
         sum : out std_logic;
         carry : out std_logic);
end half_adder;

architecture Behavioral of half_adder is
  -- declaration
begin
  -- description of behavior
end Behavioral;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity half_adder is
    Port ( x ,y: in std_logic;
           sum,carry : out std_logic);
end half_adder;

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;