ASIC Design using Verilog/VHDL

(EC9036)

VHDL - Introduction

- VHDL is a hardware description language. The code describes the behavior or structure of an electronic circuit, from which a compliant physical circuit can be inferred by a compiler.
- Its main applications include synthesis of digital circuits onto CPLD/FPGA (Complex Programmable Logic Device/Field Programmable Gate Array) chips and layout/mask generation for ASIC (Application-Specific Integrated Circuit) fabrication.
- VHDL stands for VHSIC (Very High Speed Integrated Circuits) Hardware Description Language, and resulted from an initiative funded by the U.S. Department of Defense in the 1980s.
- VHDL allows circuit synthesis as well as circuit simulation

Synthesis and Simulation

- Synthesis is the translation of a source code into a hardware structure that implements the intended functionality, while the Simulation is a testing procedure to ensure that such functionality is indeed achieved by the synthesized circuit.
- EDA Tools

--From Xilinx: ISE (XST for synthesis, ISE Simulator for simulation)

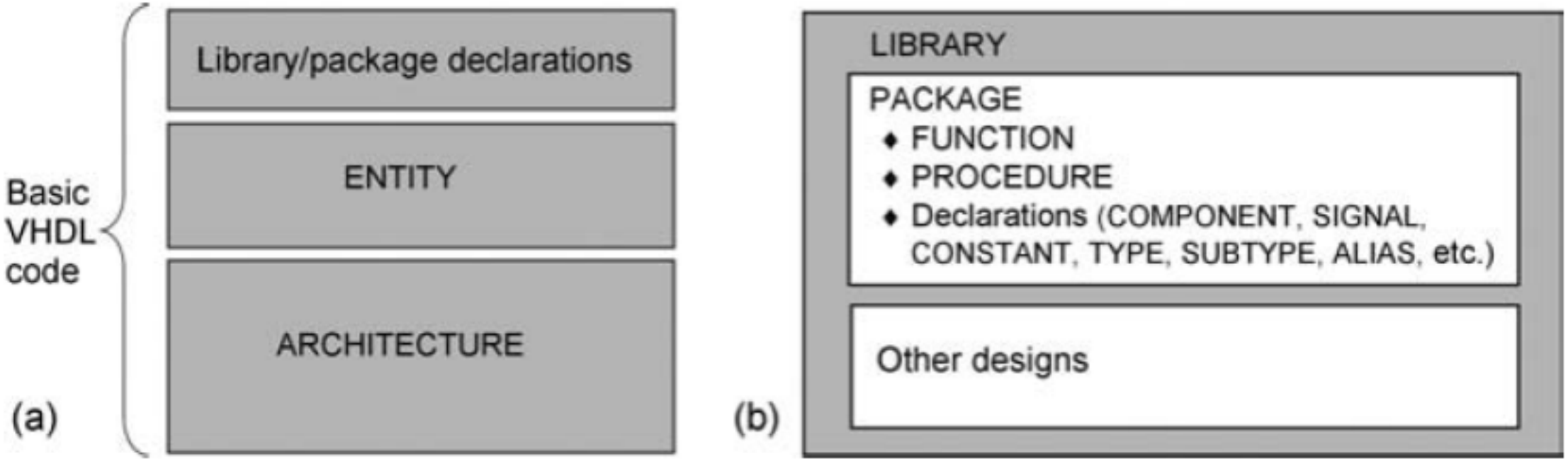
--From Synopsys: Design Compiler Ultra and Premier (synthesis), VCS (simulation), etc.

- -- From Mentor Graphics: Precision RTL and Leonardo Spectrum (synthesis), ModelSim (simulation)

Synthesis of Digital Systems

- Synthesis
- "abstract high-level specification" to "detailed implementation“
- Analogy with Compiler
- Both translate high-level specification to low-level
- **Outputs are different**
- Compiler output: instruction stream + operands
- Synthesis output may be a combination of different Hardware components (ALU, MUX, Registers, Memories, etc.)

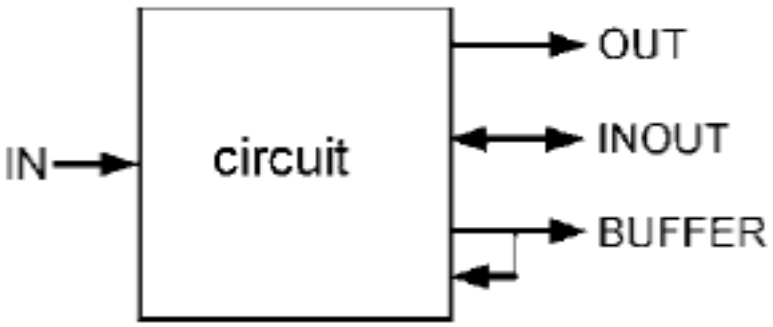
VHDL code structure



(a) Fundamental sections of a VHDL code; (b) Fundamental parts of a library.

ENTITY and ARCHITECTURE

- The main part of an ENTITY is PORT, which is a list with specifications of all input and output ports (pins) of the circuit. Entity has interface information only. It does not define any functionality.



VHDL port modes;

- ARCHITECTURE contains a description of how the circuit should function, from which the actual circuit is inferred.
- For a given entity, we may have different architectures.
  -- different levels of detail
  -- different views

```
LIBRARY library_name;
USE library_name.package_name.all;
```

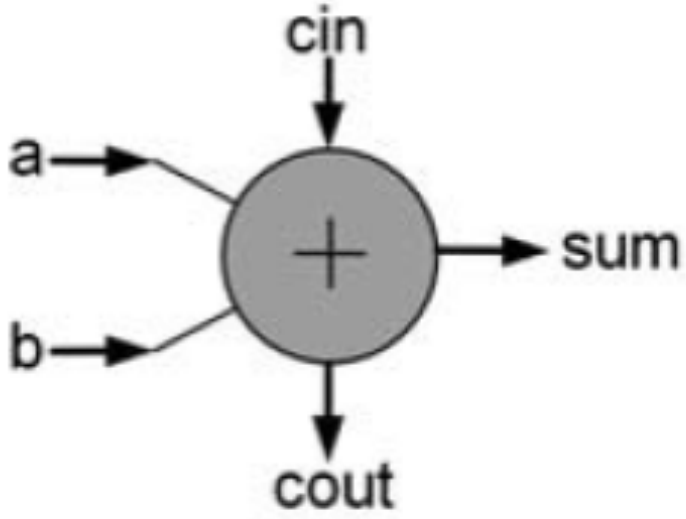
```
ENTITY entity_name IS
   PORT (
      port_name: port_mode signal_type;
      port_name: port_mode signal_type;
      ...);
END [ENTITY] [entity_name];
```

```
ARCHITECTURE architecture_name OF entity_name IS
  [architecture_declarative_part]
BEGIN
  architecture_statements_part
END [ARCHITECTURE] [architecture_name];
```

Important points

- VHDL is not case sensitive, except possibly for the STD_ULOGIC symbols ('Z', 'X', etc.).
- VHDL code is inherently concurrent (contrary to regular computer programs (HLL codes), which are sequential).
- Remember that in a concurrent code the order of the statements does not matter. For example, if a code uses three concurrent statements, called stat1, stat2, and stat3, then any of the following sequences will render the same physical circuit: {stat1,stat2,stat3}={stat2,stat3,stat1} ={stat3,stat2,stat1}.

Example : Full adder circuit



a	b	cin	sum	cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

```
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity full_adder is
33      Port ( a : in  STD_LOGIC;
34             b : in  STD_LOGIC;
35             cin : in  STD_LOGIC;
36             sum:  out STD_LOGIC;
37             cout : out  STD_LOGIC);
38  end full_adder;
39
40  architecture arch1 of full_adder is
41
42  begin
43  sum <= a xor b xor cin;
44  cout <= (a and b) or (b and cin) or (cin and a);
45  end arch1;
```

```
14  library IEEE;
15  use IEEE.STD_LOGIC_1164.ALL;
16
17  entity full_adder2 is
18      Port ( a : in  STD_LOGIC;
19             b : in  STD_LOGIC;
20             cin : in  STD_LOGIC;
21             sum : out  STD_LOGIC;
22             cout : out  STD_LOGIC);
23  end full_adder2;
24  architecture arch2 of full_adder2 is
25  signal input: std_logic_vector (2 downto 0);
26  signal output: std_logic_vector (1 downto 0);
27  begin
28  input <=a & b &cin;
29  sum <= output(1);
30  cout <= output(0);
31  with input select
32  output <= "00" when "000",
33            "10" when "001",
34            "10" when "010",
35            "01" when "011",
36            "10" when "100",
37            "01" when "101",
38            "01" when "110",
39            "11" when "111",
40            "XX" when others;
41  end arch2;
```

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  entity full_adder3 is
23      Port ( a : in  STD_LOGIC;
24             b : in  STD_LOGIC;
25             cin : in  STD_LOGIC;
26             sum : out  STD_LOGIC;
27             cout : out  STD_LOGIC);
28  end full_adder3;
29
30  architecture arch3 of full_adder3 is
31  COMPONENT half_adder
32     PORT(x,y : IN std_logic;
33          u,v : OUT std_logic);
34     END COMPONENT;
35  COMPONENT orgate
36     PORT(x,y : IN std_logic;
37           z: OUT std_logic);
38     END COMPONENT;
39  signal s1,c1,c2: STD_LOGIC;
40  begin
41  ha1: half_adder port map(a,b,s1,c1);
42  ha2: half_adder port map(s1,cin,sum,c2);
43  orgate1: orgate port map(c1,c2,cout);
44  end arch3;
```