VHDL - Code Structure

Entity, Architecture, Configuration

.vhd File Structure

A standalone file of VHDL code is composed of at least three fundamental sections:

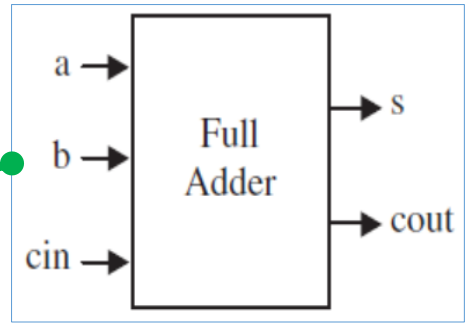
- **LIBRARY** declarations: Contains a list of all libraries to be used in the design. For example: ***ieee, std, work***, etc.

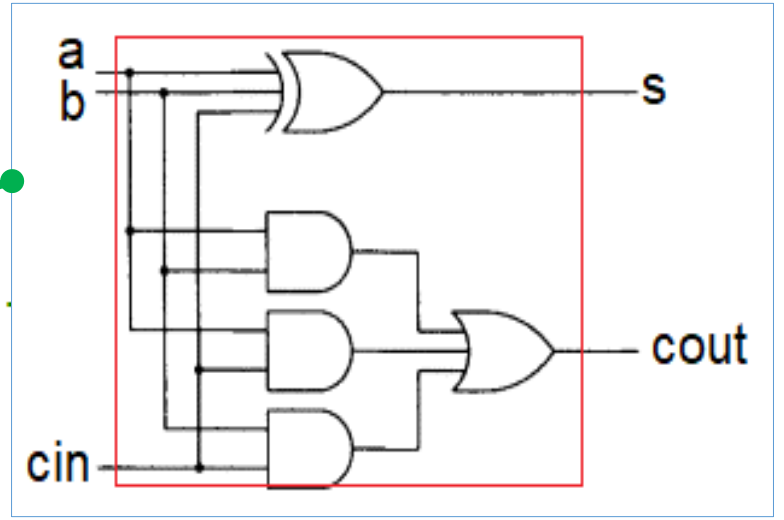
- **ENTITY**: Specifies the I/O pins of the circuit.

- **ARCHITECTURE**: Contains the VHDL code proper, which describes how the circuit should behave and function.

.vhd File Structure

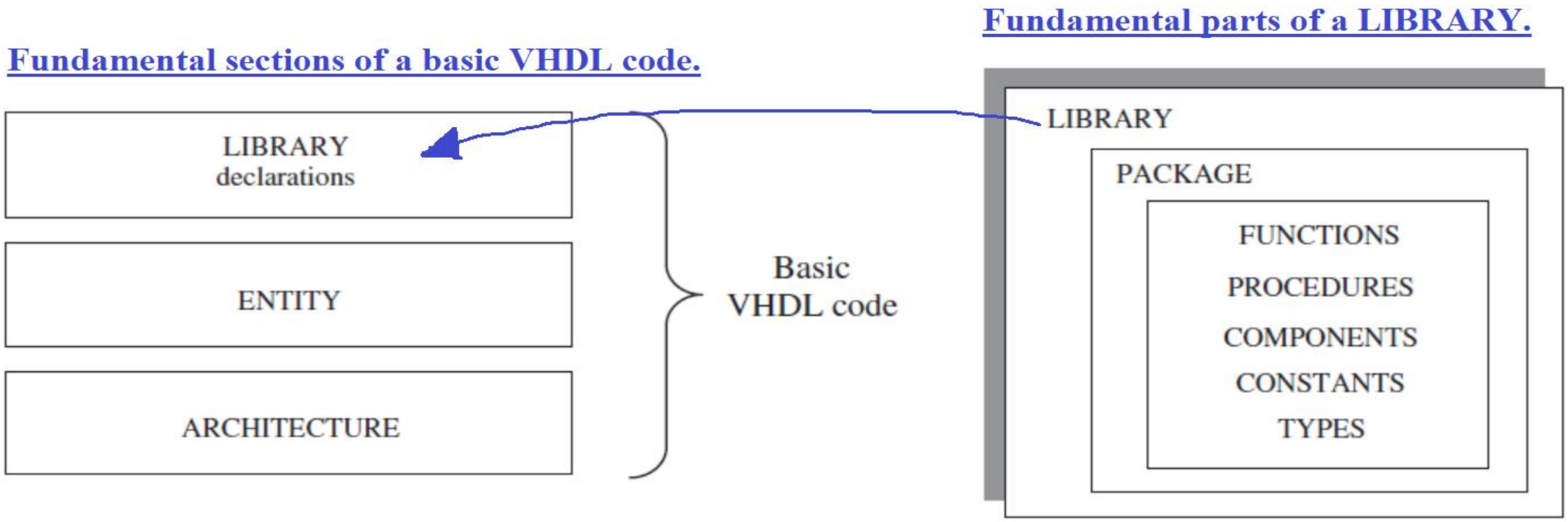
```
-- Library Definition
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
---------- Entity Definition ----------------
ENTITY FA IS
     PORT (a, b, cin: IN std_logic;
                s, cout: OUT std_logic);
END FA;
---------- Architecture Definition --------
ARCHITECTURE dataflow OF FA IS
BEGIN
     s <= a XOR b XOR cin;
     cout <= (a AND b) OR (a AND cin) OR(b AND cin);
END dataflow;
```





Library Declarations

- To declare a **LIBRARY** (make it visible to the design) two lines of code are needed, one containing the name of the library, and the other a use clause, as shown before.



Entity

Entity defines the input and output of our design as a “black box” view.

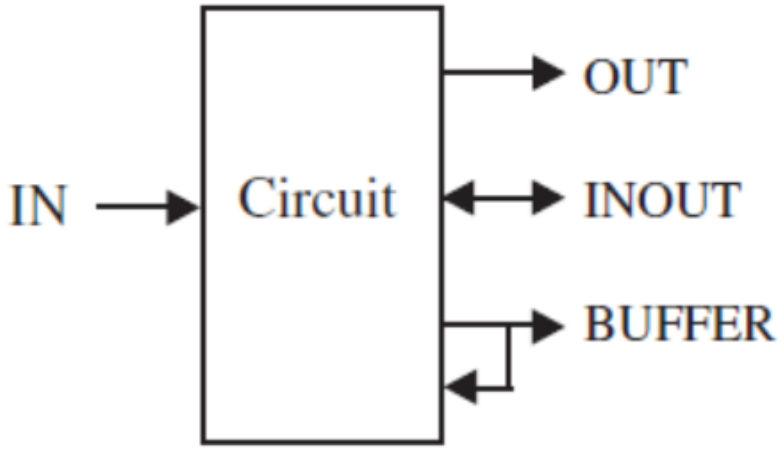
```
entity NAME_OF_ENTITY is
   generic (generic_declarations);
   port (signal_names: mode type;
         signal_names: mode type;
                :
         signal_names: mode type);
end NAME_OF_ENTITY ;
```

Entity - mode of the signals (PORTs)

mode: is one of the reserved words to indicate the signal direction:

- **in** – input signal
- **out** – output signal
- **inout** – usually used for bi-directional bus
- **buffer** - output signal and can be read

  All are read by other entities



***type*:** example of signal type.

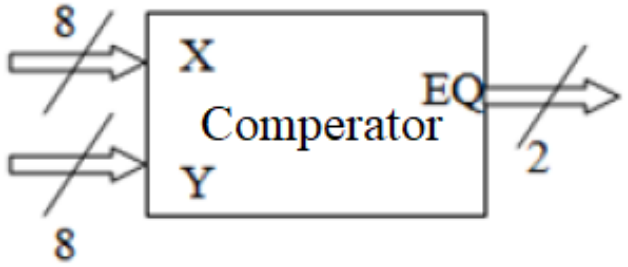
- ***bit*** *– Boolean 1 or 0*
- ***bit_vector*** *– Boolean vector*

generic: determine the architecture local constants.

generic (constant values of types: NATURAL, POSITIVE, INTEGER, STRING);

Entity - Examples


```
1    ENTITY nd2 IS
2      PORT(a,b:IN bit; z:OUT bit);
3    END nd2;
```




```
1    ENTITY comp IS
2      GENERIC(n: integer:=8);
3      PORT(x,y:IN bit_vector(n-1 downto 0); --8 bit buses
4           eq:OUT bit_vector(1 downto 0)); --2 bit buses
5    END nd2;
```

Architecture

- The **ARCHITECTURE** is a description of how the circuit should behave.
- An **ARCHITECTURE** must be associated to only single **ENTITY.**

```
architecture architecture_name of entity_name is
    [ component declaration ]
    [ signal declaration ]
begin
    [ design logic - the code part]
end  architecture_name ;
```

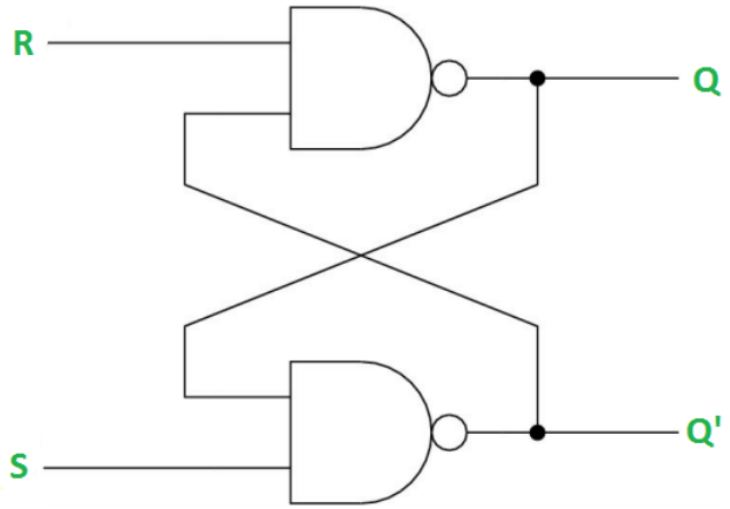
Architecture – Example1


```
1    ENTITY nd2 IS
2      PORT(a,b:IN bit; z:OUT bit);
3    END nd2;
4
5
6    ARCHITECTURE dtf_nd2 OF nd2 IS
7      BEGIN
8        Z<= a nand b;
9    END dtf_nd2;
```

Architecture – Example2

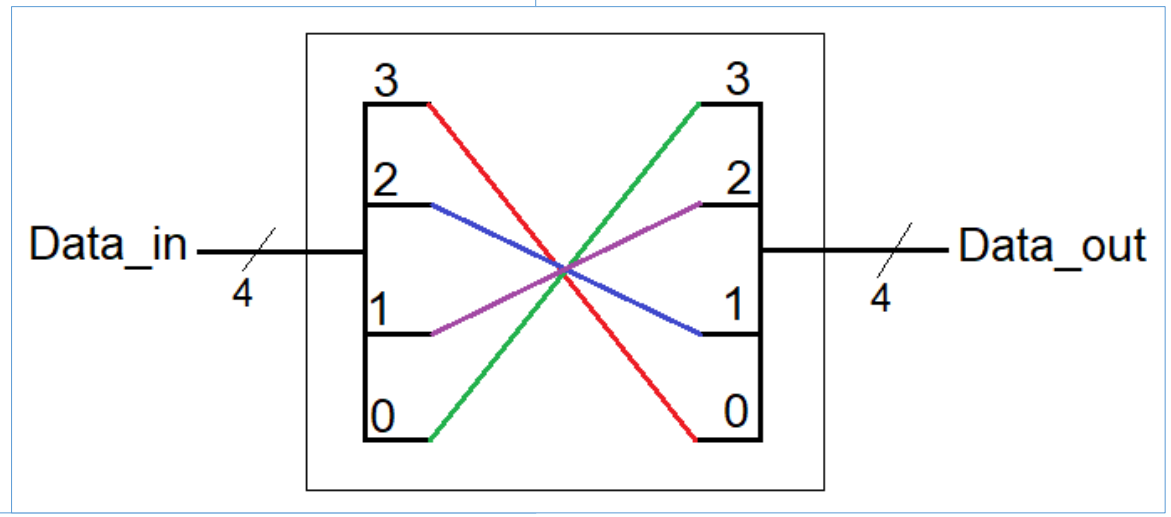
```
 1   LIBRARY IEEE;
 2   USE IEEE.STD_LOGIC1164.ALL;
 3
 4   ENTITY rs_ff IS
 5     PORT(set,reset:IN std_logic;
 6          q,q_not :BUFFER std_logic);
 7   END rs_ff;
 8
 9
10   ARCHITECTURE bottom_up OF rs_ff IS
11     BEGIN
12       q <= reset nand q_not;
13       q_not <= set nand q;
14   END bottom_up;
```

RS Latch



Architecture – Example3

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC1164.ALL;
3
4   ENTITY switch IS
5     GENERIC(size1: NUATURAL :=4; size2: NUATURAL :=3);
6     PORT(Data_in:IN std_logic_vector(size1-1 downto 0);
7          Data_out:BUFFER std_logic_vector(size1-1 downto 0));
8   END switch;
9
10
11  ARCHITECTURE generics OF switch IS
12    BEGIN
13      Data_out(size1-1) <= Data_in(0);
14      Data_out(0) <= Data_in(size1-1);
15      Data_out(size2-1) <= Data_in(1);
16      Data_out(1) <= Data_in(size2-1);
17  END generics;
```



Configuration – Simulation Environment

- Configuration is a simulation design unit which flexes the design process.
- In order to associate one **ARCHITECTURE** from several options, we use Configuration.

```
configuration configuration_name of entity_name is
   for architecture_name
   end for;
end configuration configuration_name ;
```