University of Leicester

Programmable Electronics using FPGAs

This week

- VHDL essentials - data objects and their types
- Hierarchical design
- Abstraction levels – Structural modelling
- Component instantiation
- Connecting ports
- Lab work 1: Design a 1-bit Full Adder
- Lab work 2: Design a 2-bit Full Adder

Data objects

- Three types of data objects are used in VHDL:
  - Signals
  - Constants
  - Variables
- Signals are most commonly used to describe the logic circuit. They represent the wires of the circuit.
- Constants are used to define a value which cannot be changed.
- Variables are usually used to store computational data within a procedural block. They do not necessarily represent a wire in a circuit.

Data object values and numbers

- bit values: '0', '1'
- boolean values: TRUE, FALSE
- integer values: 152, 1, 9 etc
- std_logic values: U, X, -, Z, H, L, 0, 1:
  - **U** = uninitialized
  - **X** = **-** = don’t care
  - **Z** = undefined voltage range
  - **H/L** = weak 0 or 1 – can’t be trusted in VHDL!

- std_logic_vector values: b“00101101” , b“1101” , x”A07B” etc

STD_LOGIC and STD_LOGIC_VECTOR

- STD_LOGIC is used for single bit signals
- STD_LOGIC_VECTOR is used for multi-bit signals (words).

  For example :

  ```
  signal X std_logic;
  signal Y std_logic_vector(3 downto 0);
  signal Z std_logic_vector(0 upto 5);
  ```

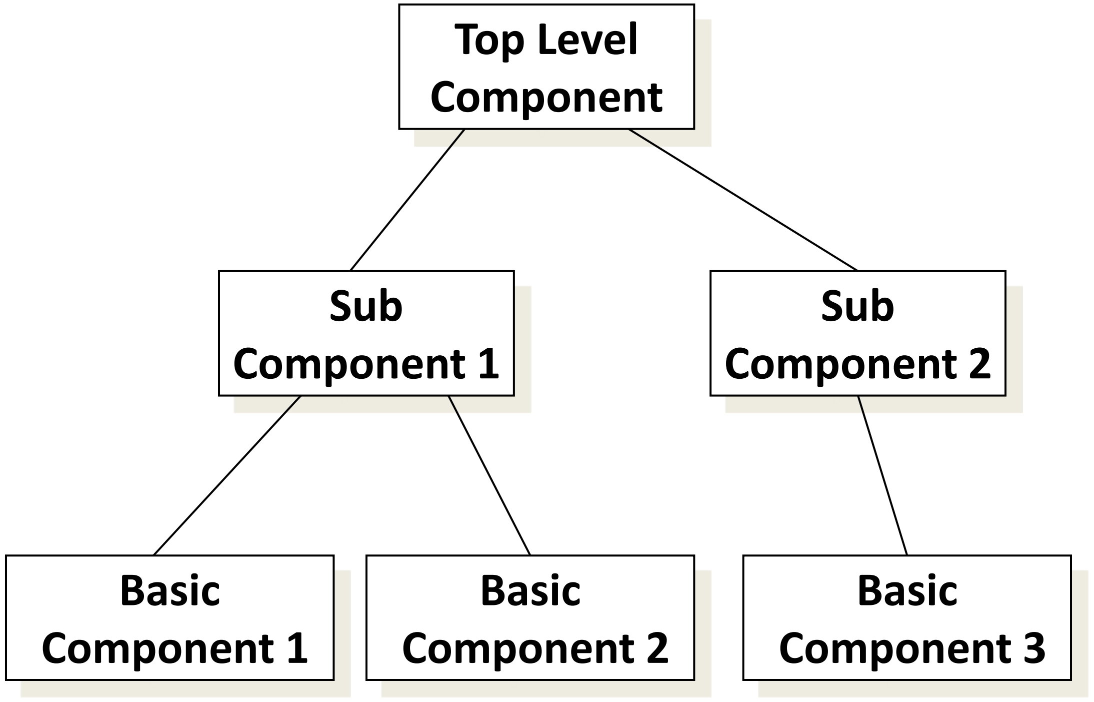
- STD_LOGIC_VECTOR is also bit accessible.
  - Individual bits can be accessed, just like an array.
- The least significant bit is the one indexed **0** in the declaration.

  For example : Y <= “0101”;

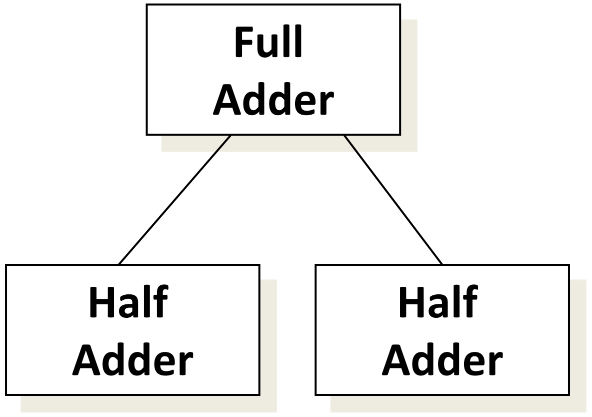
  Here Y(0) = 1, Y(1) = 0, Y(2) = 1 and Y(3) = 0

Hierarchical design

- We often build a digital system using existing components.



For example : A full adder consists of two half adders



Abstraction levels: Structural Modelling

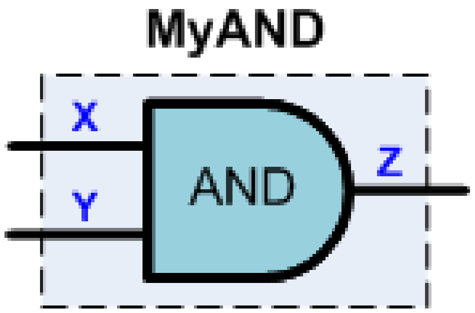
- Structural modelling is used to connect (instantiate) existing components with each other.
- It describes the arrangement and interconnections of components via their ports.
- Structural modelling is performed in two steps:
    1. Component instantiation
    2. Connecting ports

Step 1: Component instantiation

- Entity/architecture pairs can be declared as a “component”
- This allows us to treat them as black boxes in larger designs.

For example : AND gate

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY MyAND IS
PORT(
   X,Y  : IN STD_LOGIC;
      Z  : OUT STD_LOGIC
      );
END MyAND;

ARCHITECTURE dataflow of MyAND IS
Begin
     Z <= X AND Y;
end dataflow;
```



```
COMPONENT MyAND IS
   PORT (
         X : IN STD_LOGIC;
         Y : IN STD_LOGIC;
         Z : OUT STD_LOGIC
      );
END COMPONENT;
```

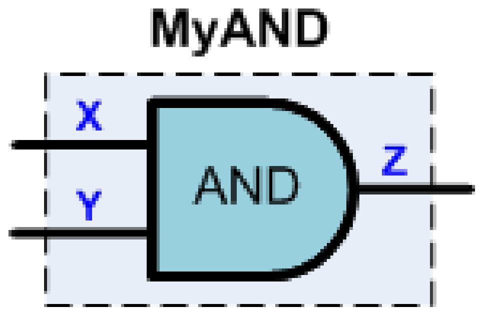
Step 1: Component instantiation

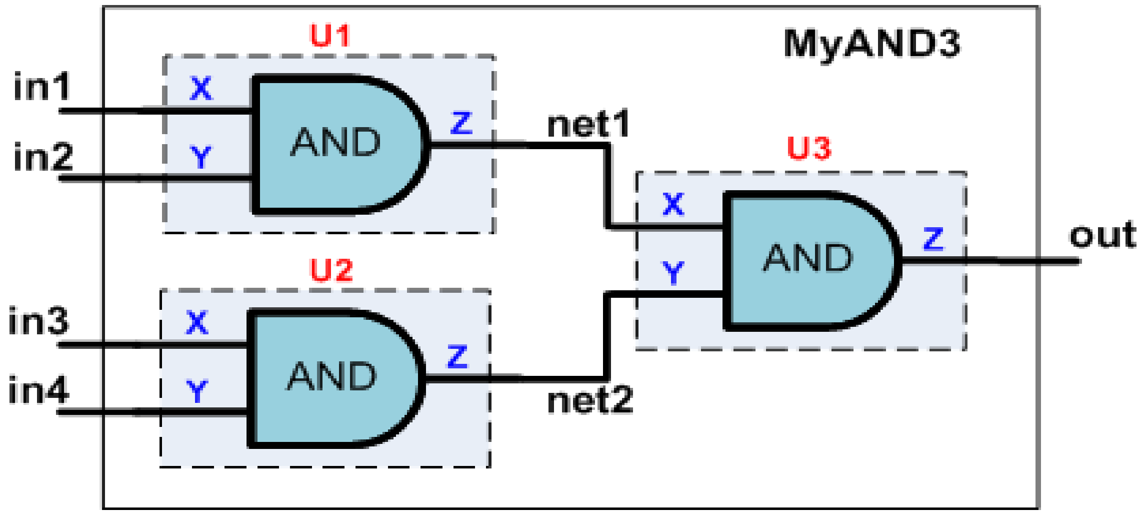
- VHDL models consist of hierarchy of component instances.

For example:

MyAND3 consists of 3 MyAND instances U1, U2, and U3 with the pins connected together:

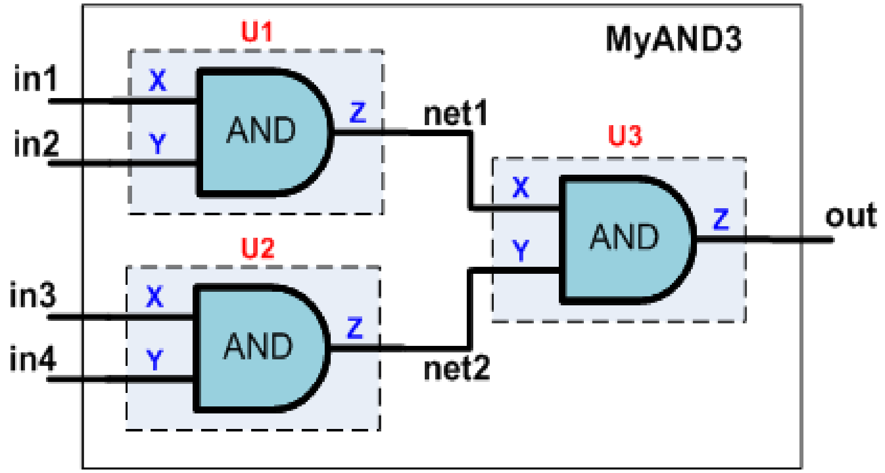
```
U1: MyAND port map (Port connections…);
U2: MyAND port map (Port connections…);
U3: MyAND port map (Port connections…);
```





Step 2: Connecting ports

- Components can be connected together by specifying the component names of each specific pin



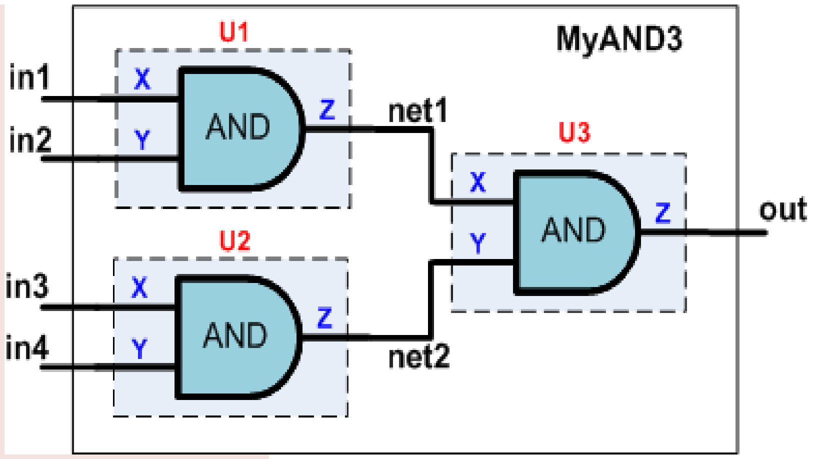
For example :

MyAND3 consists of 3 MyAND gates U1, U2, and U3 with the pins connected together:

```
U1: MyAND port map (X => in1, Y => in2, Z => net1);
U2: MyAND port map (X => in3, Y => in4, Z => net2);
U3: MyAND port map (X => net1, Y => net2, Z => out);
```

Example: Hierarchical circuit design

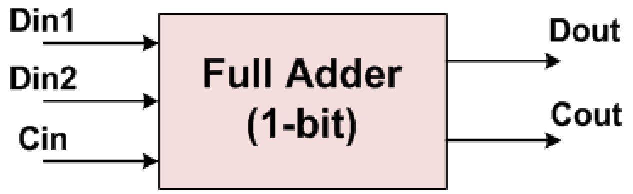
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY MyAND3 IS
  PORT(
        in1,in2,in3, in4 : IN STD_LOGIC;
                    out : OUT STD_LOGIC);
 END MyAND3;

ARCHITECTURE structural of MyAND3 IS
COMPONENT MyAND IS
   PORT (
         X : IN STD_LOGIC;
         Y : IN STD_LOGIC;
         Z : OUT STD_LOGIC);
END COMPONENT;
signal net1, net2 : STD_LOGIC;
Begin

   U1: MyAND port map (X => in1, Y => in2, Z => net1);
   U2: MyAND port map (X => in3, Y => in4, Z => net2);
   U3: MyAND port map (X => net1, Y => net2, Z => out);

end structural;
```

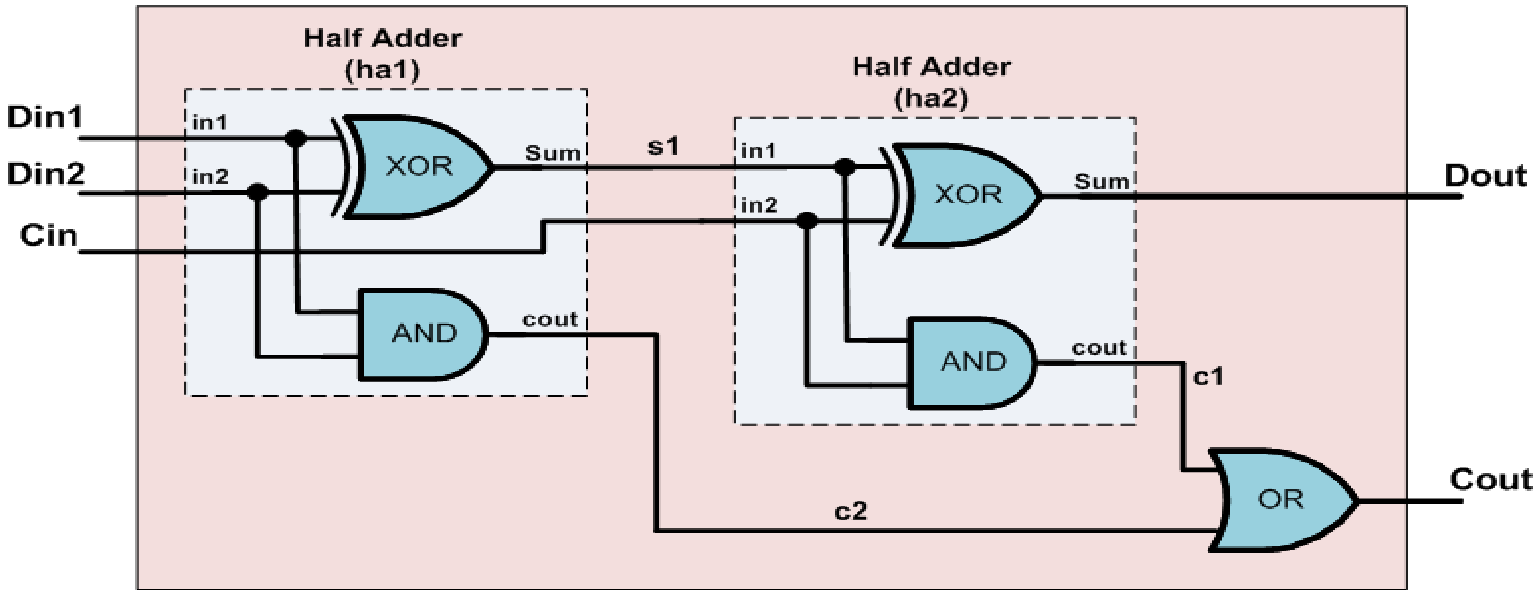


Lab work 1: Designing a 1-bit full adder

- A full adder consists of two half adders and an OR gate.
- These circuits operate side by side.

Example: 1-bit full adder





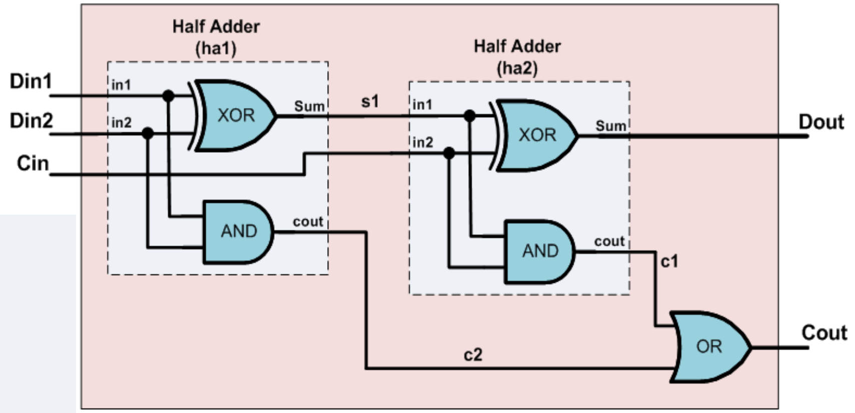
Lab work 1: Designing a 1-bit full adder

- Use your half adder design as a component.
- You need one concurrent statement and two instantiations in an architecture.
  - Concurrent statement is for OR gate.
  - One instantiation per a half adder component.

Hints: Use dataflow level for OR gate and structural level for instantiation of two half adders

```
-- For OR gate
Cout <= c1 OR c2;
```

```
-- Instantiation of a half adder
ha1: HalfAdder port map(
                           in1 => Din1,
                           in2 => Din2,
                           Sum => s1,
                           cout => c2
                          );
```

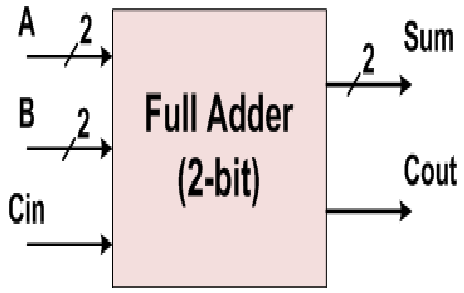


Lab work 2: Designing a 2-bit full adder

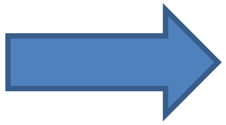
- A 2-bit full adder can designed using two 1-bit full adders.
- Follow the same rule as for 1-bit full adder design.

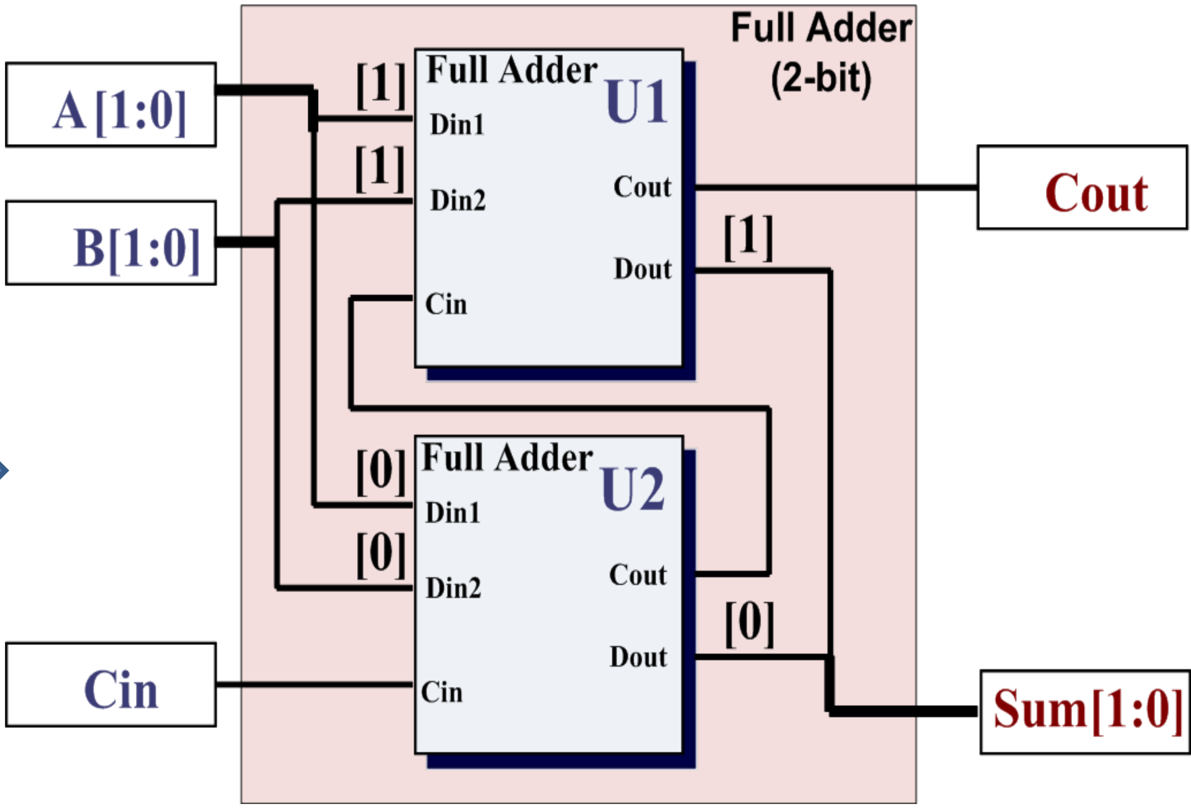
Example:

2-bit full adder



A
2
B
2
Cin
Full Adder
(2-bit)
2
Sum
Cout





A[1:0]
B[1:0]
Cin
Full Adder
(2-bit)
[1]
[1]
Full Adder
U1
Din1
Din2
Cout
Dout
Cin
[0]
[0]
Full Adder
U2
Din1
Din2
Cout
Dout
Cin
[1]
[0]
Cout
Sum[1:0]