SYSTEM MODELLING AND SYNTHESIS WITH HDL

DTEK0078

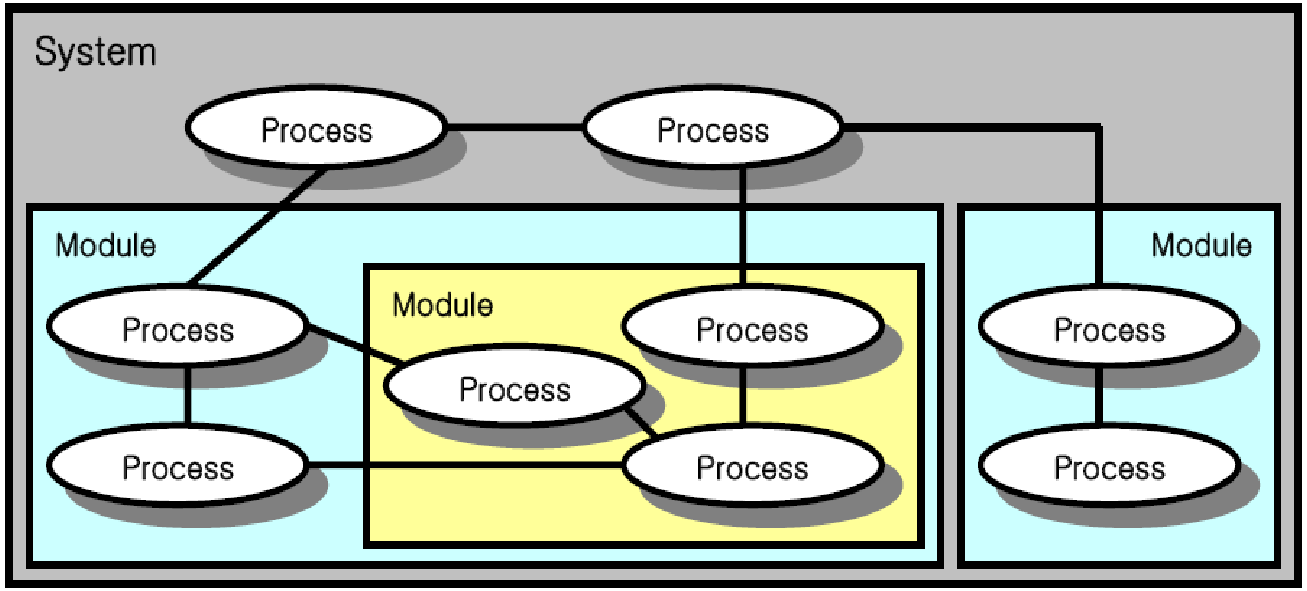
2023 Lecture 3

UNIVERSITY OF TURKU

Testing

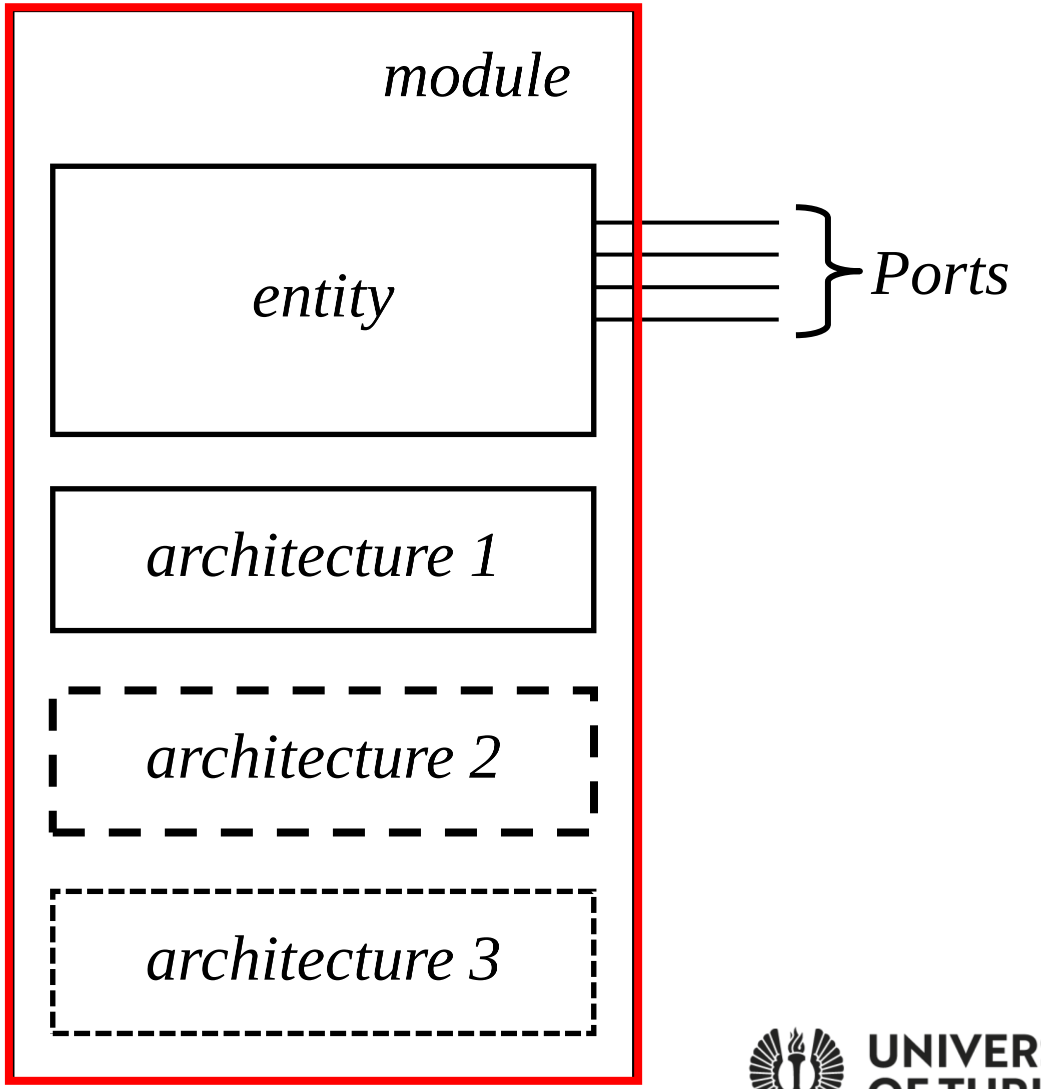
Basic VHDL Concepts

- Entities
- Architectures
- Packages



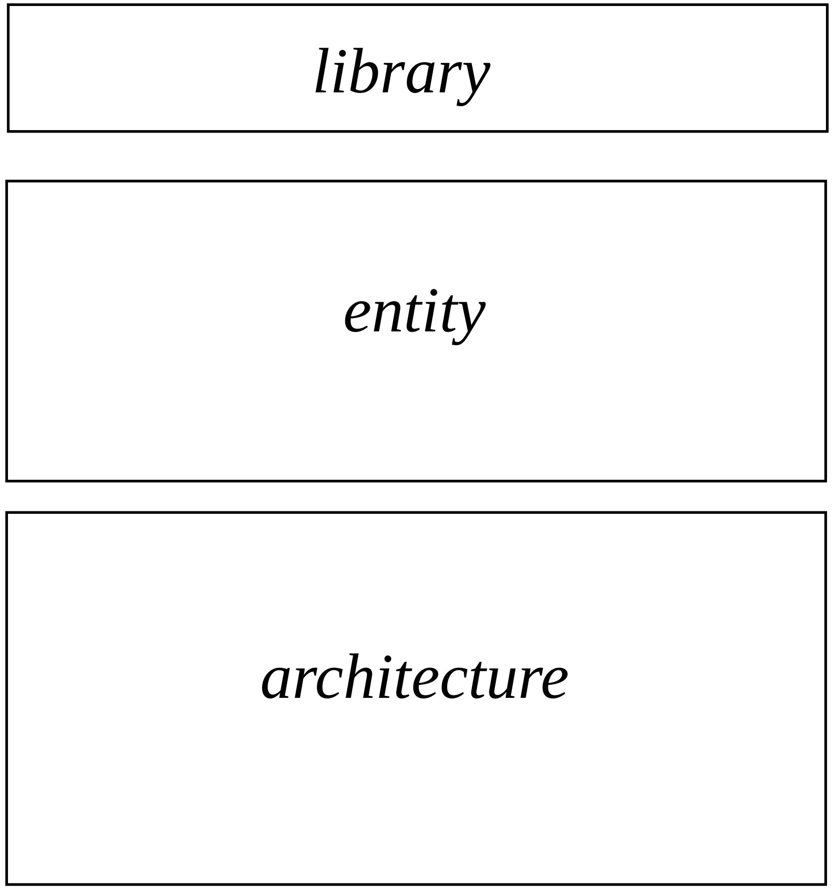
Design Entity

- Entity
  - Defines the interface of the module
- Several Architecture
  - Different implementations of the model
  - Same entity



Libraries and Packages

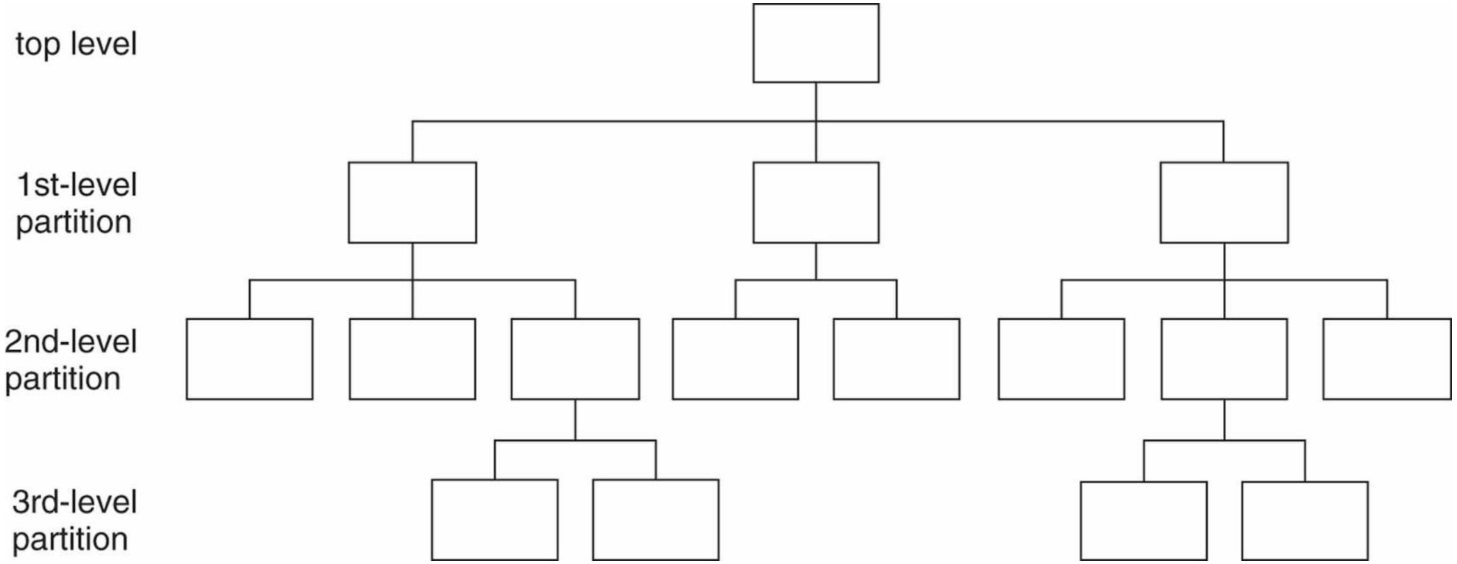
```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

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

architecture myadder of half_adder is
begin
       sum <= x xor y;
       carry <= x and y;
end myadder;
```



library

entity

architecture



Hierarchical Design

- Top-down design
- Each module may itself be partitioned to further reduce its complexity

Advantages of Hierarchical Design

01

Easier design complexity management

- Divided-and-conquer strategy
- System can be implemented in stages (concurrently)

02

Reuse of modules

- Predefined modules or third-party designs

03

Simple verification

Advantages of Hierarchical Design

BY RESTRICTING THE POSSIBLE CONNECTIONS AMONG MODULES, WE EXCHANGE FLEXIBILITY FOR CLARITY AND MAINTAINABILITY

A HIERARCHY SIMPLIFIES THE CONNECTIONS IN A SYSTEM BY RESTRICTING THE COMMUNICATION PATHS AMONG ITS MODULES

Hierarchy Improves Several Design Qualities

01

Enhances comprehensibility by providing an organized approach to understanding the system

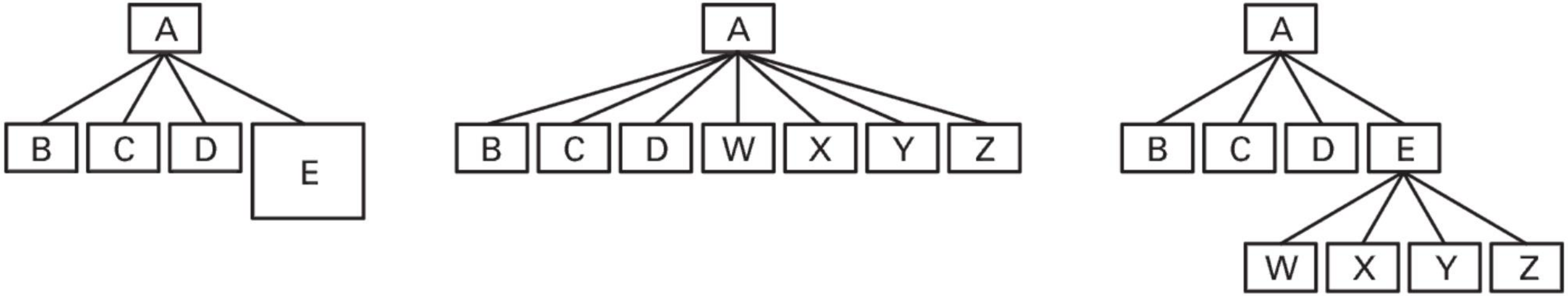
02

Reduces intermodule dependence, which in turn improves modifiability and reusability

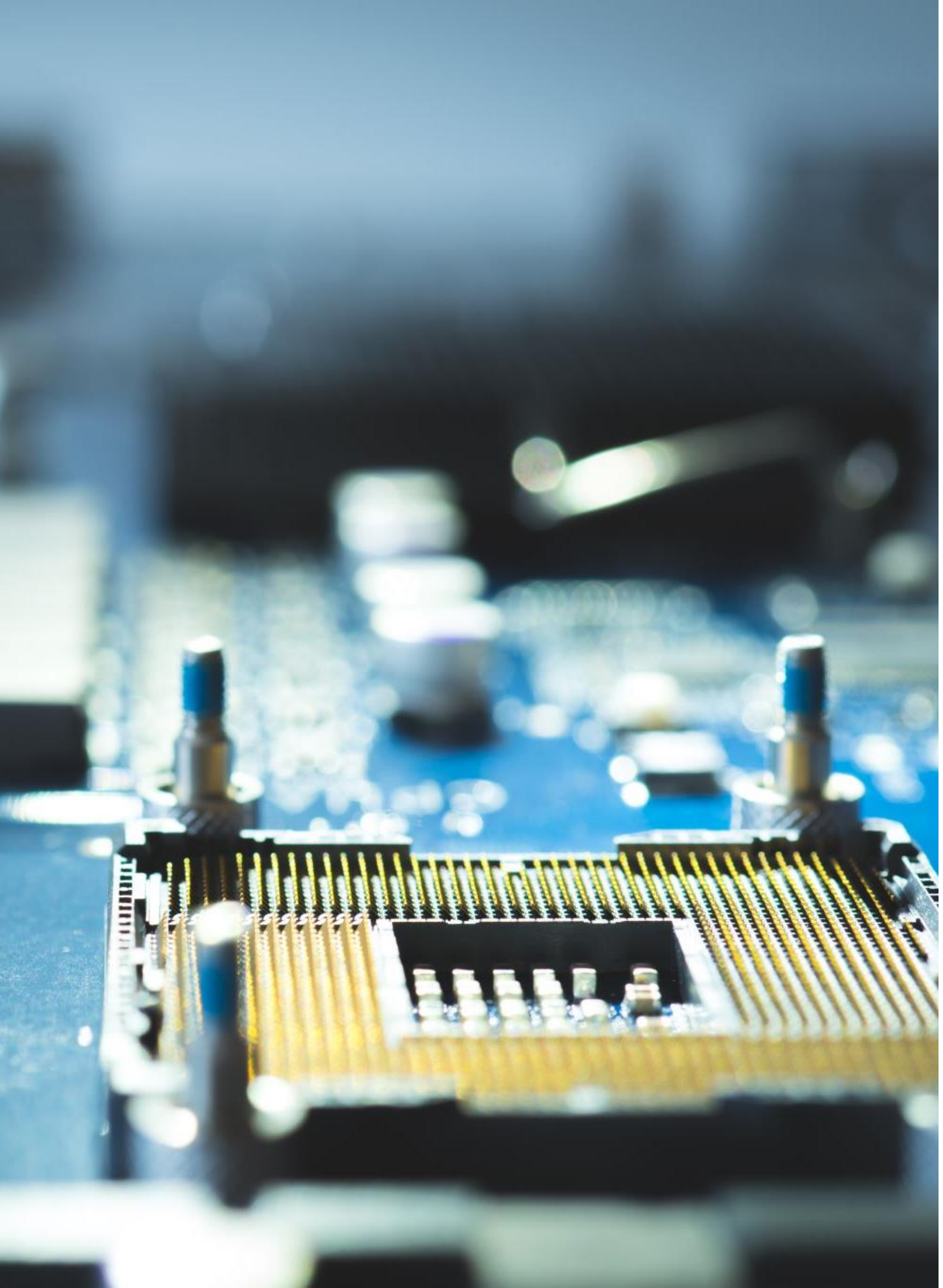
03

Provides a way to reduce the average module size in a system

Average module size in a system



- Each module may itself be partitioned to further reduce its complexity

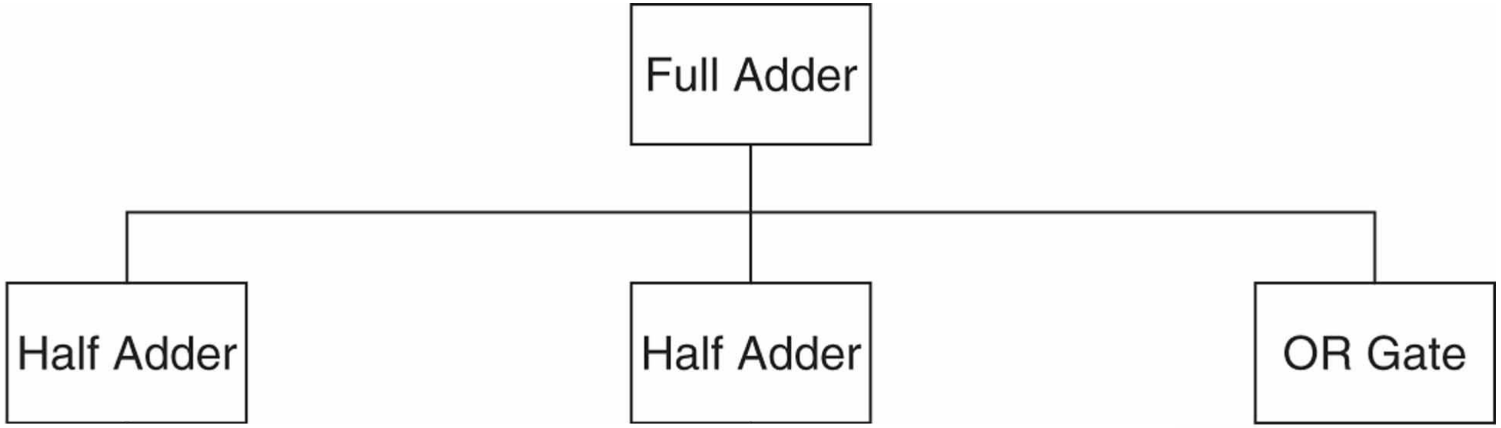


Components

UNIVERSITY OF TURKU

Components

- Component instantiation defines a subcomponent of the desing entity in which it appears



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

```
component half_adder
    port(x,y: in  std_logic;
         sum,carry: out std_logic);
    end component;
```

Components in Use

```
entity full_adder is
port(  x,y, carry_in: in std_logic;
     sum, carry_out: out std_logic);
end full_adder;

architecture structural of full_adder is

    signal s1, s2, s3: std_logic;

    -- Component declarations
    component half_adder
        port(x,y: in  std_logic;
               sum, carry: out std_logic);
    end component;

begin
    -- Component instantiations
    hf1: entity half_adder(dataflow) port map (x, y, s1, s2);
    hf2: entity half_adder(dataflow) port map (s1, carry_in, sum, s3);

    carry_out <= s2 or s3;

end structural;
```

Component Instantiations

- Entity and architecture

```vhdl
ha1: entity half_adder(dataflow)
        port map (x, y, s1, s2);
```

- Library, entity and architecture

```vhdl
ha1: entity work.half_adder(behavioural)
        port map (x, y, s1, s2);
```

- Entity without architecture and library

```vhdl
ha1: half_adder port map (x, y, s1, s2);
```

Port Mapping

Declaration

```
component half_adder
    port( x,y: in  std_logic;
           sum, carry: out std_logic);
    end component;
```

Instantiation

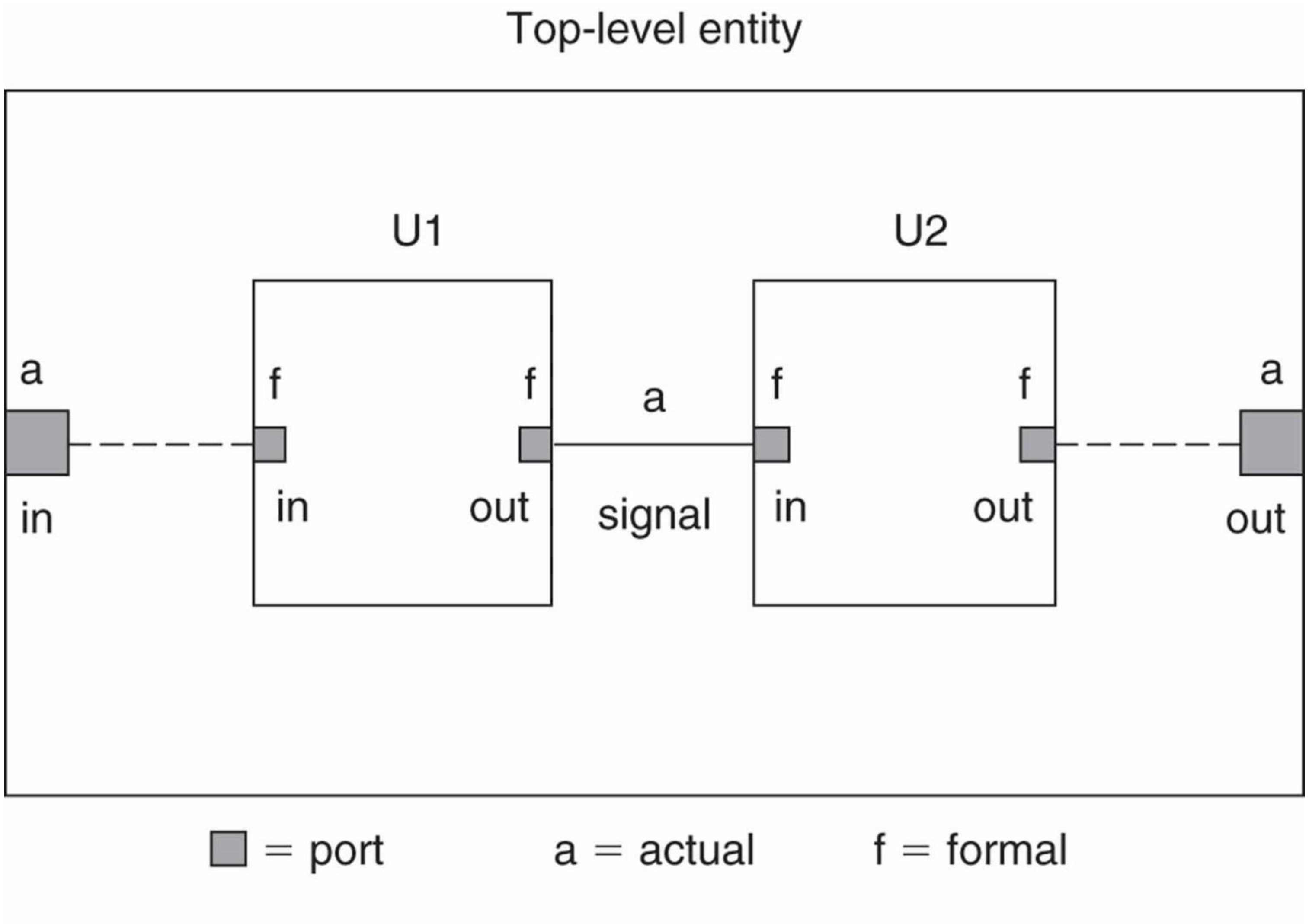
Positional association

```
ha1: entity
work.half_adder(dataflow)
    port map (x, y, s1, s2);
```

Named association

```
ha1: half_adder
    port map (x => x, y => y,
              sum => s1,
              carry  => s2);
```

Component connections

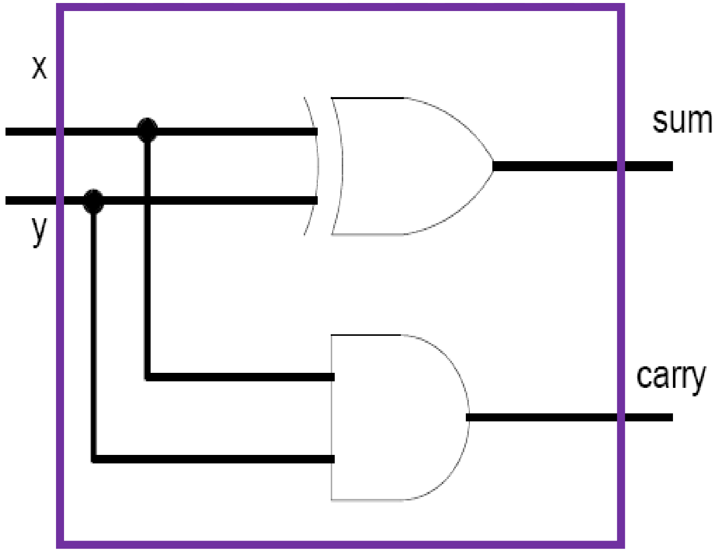


Testbenches

UNIVERSITY OF TURKU

An Example Module (recap)

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

architecture myadder of half_adder is begin
    sum <= x xor y;
    carry <= x and y;
end myadder;
```

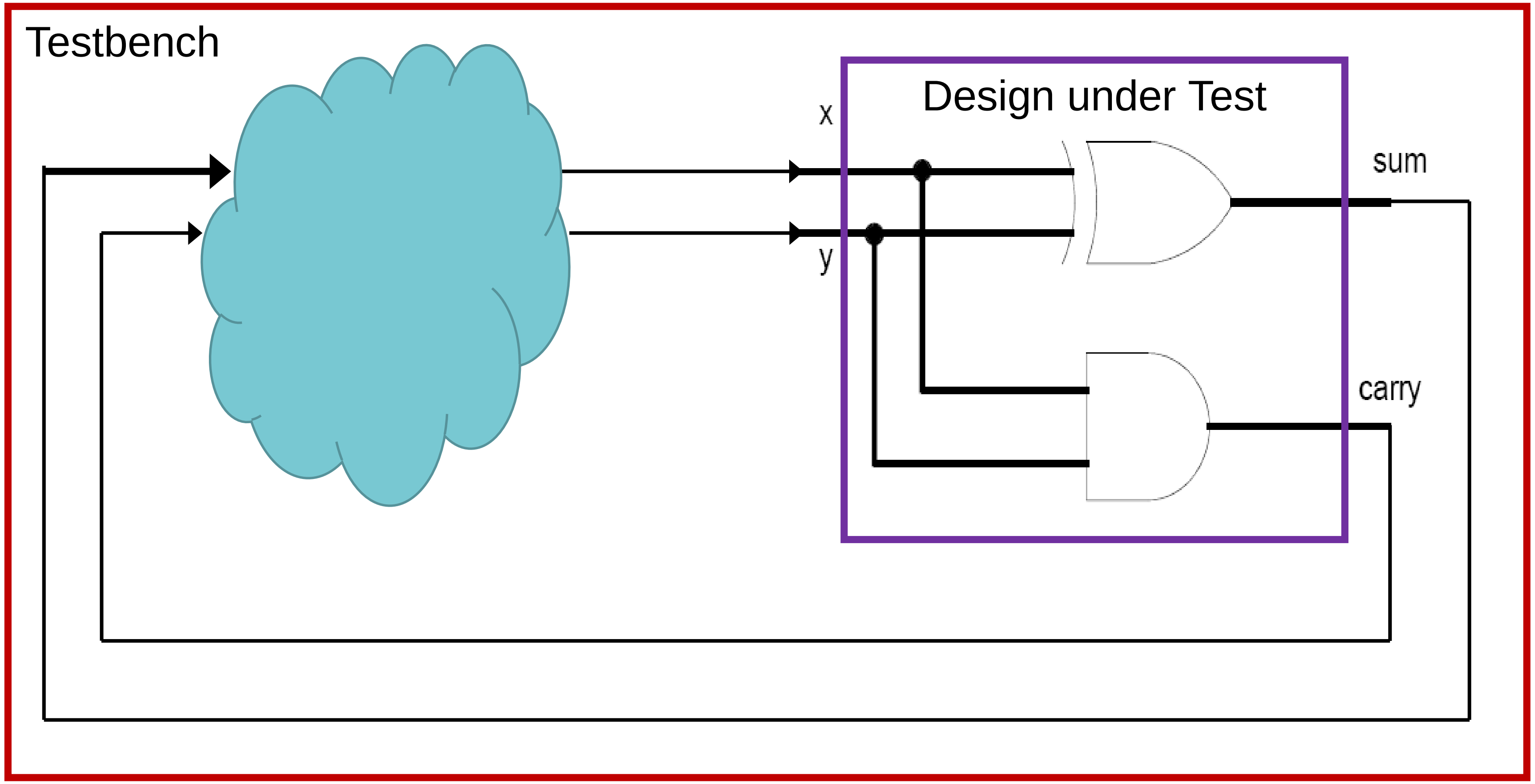


Testbenches

- A testbench is simulation code that applies some stimulus to a design, observes its response, and verifies that it behaves correctly

Testbench

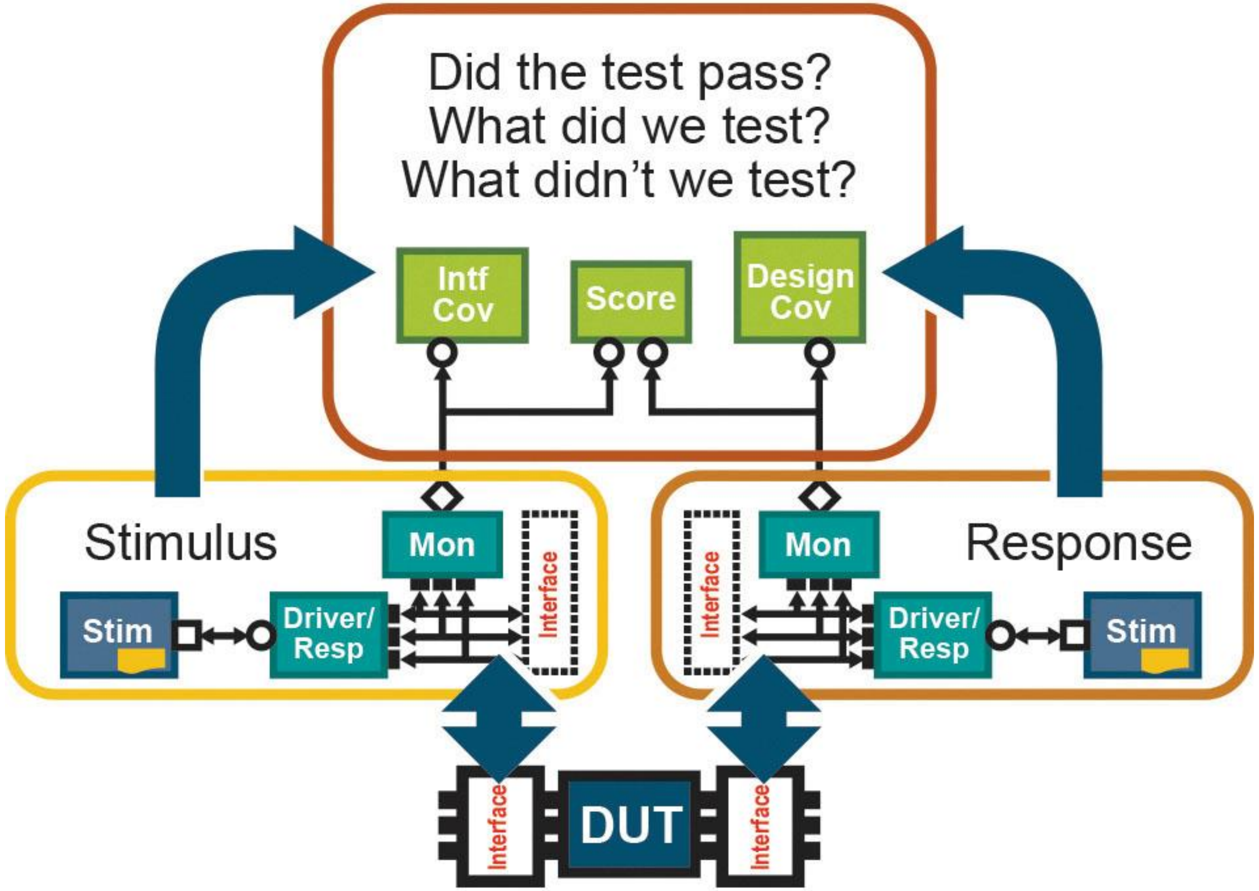
- Testbench generates input signals for Device under Test (DUT) and observes DUT's response



Testbench Structure

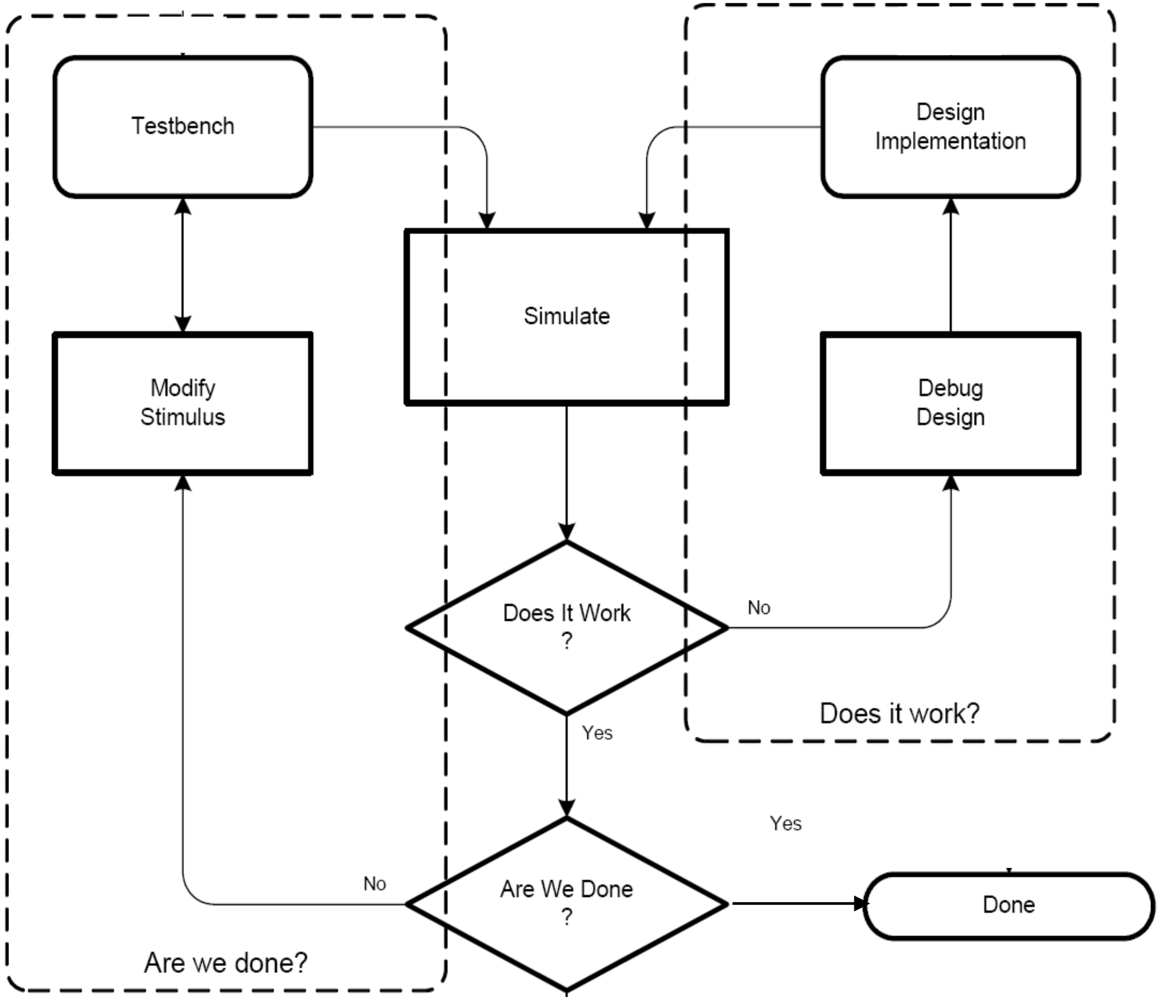
- Stimulus generator
    - Applies a sequence of predetermined stimulus values to the DUT inputs
- Response monitor
    - DUT’s output values are checked to verify that they are equivalent with the expected ones
- Self-checking testbench
    - Golden model that provides ”correct” results which are compared to DUT’s response

Testbench Structure



Verification

- Verification of a design involves answering two fundamental questions:



Simple Testbench

```
entity half_adder_tb is
end half_adder_tb;

architecture behav of half_adder_tb is
  signal t_x, t_y, t_sum, t_carry : std_logic;
  component half_adder
    port(x,y: in std_logic;
         sum,carry: out std_logic);
  end component;

begin
. . .
end behav;
```

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

Simple Testbench

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

```
entity half_adder_tb is
end half_adder_tb;

architecture behav of half_adder_tb is
  ...
begin
  dut: entity work.half_adder(dataflow)
    port map (t_x, t_y, t_sum, t_carry);
  stimulus: process is
    begin t_x <= '0’;
    t_y <= '0’;
    wait for 10 ns;
    ...
  end process stimulus;
end behav;
```

Simple Testbench

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

```
entity half_adder_tb is
end half_adder_tb;
architecture behav of half_adder_tb is
 . . .
begin
  stimulus: process is
    begin t_x <= '0’;
    t_y <= '0’;
    wait for 10 ns;
    assert ((t_sum = '0') and (t_carry = '0’))
      report "test failed for input compination 00"
      severity error;
    ...
  end process stimulus;
end behav;
```

Automatic Check

Assert Statement

- Checks whether a specific condition is true
  - A message is displayed if the condition does not hold
- Several severity levels:
  - Note
  - Warning
  - Error (*default if not otherwise stated)*
  - Failure
- For debugging, it is good to indicate the input values in a report statement if an error occured
- Can be used as a sequential or a concurrent statement

KEEP
CALM
AND
START
DEBUGGING

UNIVERSITY
OF TURKU