Budapest University of Technology and Economics
Department of Electron Devices

Lab. 2
Hierarchical design, Testbench, Concurrent statements

Osama Ali

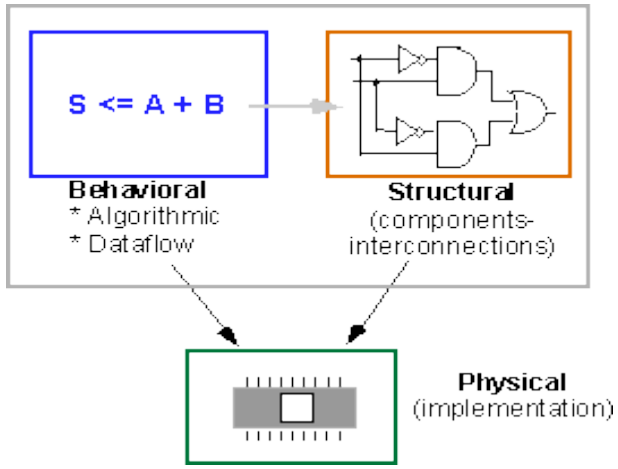
osamaalisalman.khafajy@edu.bme.hu

Ahmad Halal

ahmadhalalftesah@edu.bme.hu

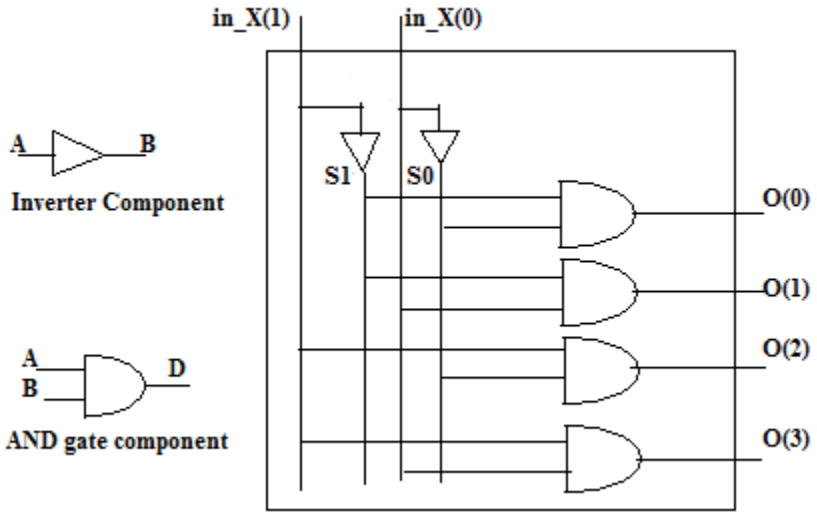
13 of May 2022

Hierarchical Modeling



- By adopting a more hierarchical design approach it is possible to reuse common elements, and segment a complex design into smaller pieces. Both of these techniques result in a more maintainable design.

- E.g.



Hierarchical Modeling

- To incorporate hierarchy in VHDL we must add **component declarations** and **component instantiations** to the model.
- **Component Represents a precompiled Entity-Architectecture Paire**
- Instantiation is selecting a component and using it as an instance in our design
- we need to declare internal signals to interconnect the components.
  - Format for Architecture body (for internal signals & hierarchy):

```
architecture architecture_name of entity_name is
signal declarations                    -- for internal signals in model
component decalarations                -- for hierarchical models
begin
    :
    component instantiations
    .
    concurrent statements
    :
end architecture architecture_name;
```

Component Declaration

- Format for component declaration:

```
component component_name is
        :
    port (signal_name(s): mode signal_type;
        :
        signal_name(s): mode signal_type);
end component component_name;
```

  - the component_name is the same as the entity_name from the model being called up.
- component declarations look just like the entity statements for the component being declared but with “component” substituted for “entity”.

Component instantiation

- The component instantiation is the actual call to a specific use of the model.
- A single component declaration can have multiple instantiations.
- each component instantiation must include a unique name (instantiation_label along with the component (component_name) being used.
- There are two methods (and formats) for connecting signals to the port of the component:

Keyword association	Positional association
instantiation_label: *component_name* : port map (*port_name* => *signal_name*, : *port_name* => *signal_name*);	*instantiation_label*: *component_name* : port map (*signal_name*, : *signal_name*);

Refreshing: Concurrent statement when else & with select

with select

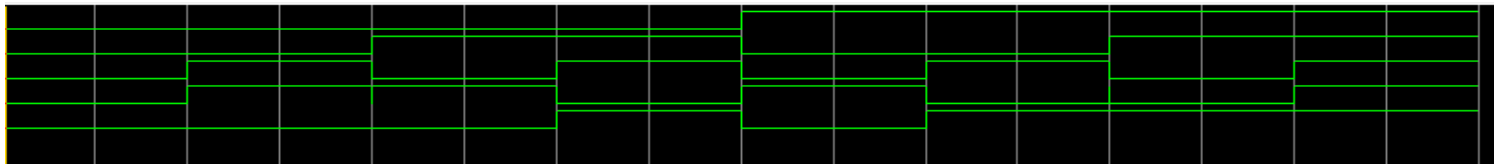
```
entity ex is
 port (a,b,c: in std_logic;
 data: in std_logic_vector (1 downto 0);
 q: out std_logic);
 end;
architecture dtf of ex is
begin
  with data select
   q <= a when “00”,
         b when “11”,
         c when others;
end;
```

when else

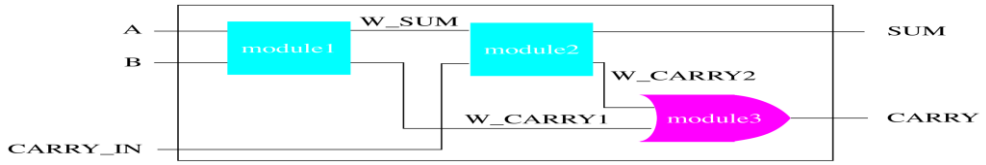
```
entity ex is
 port (a,b,c: in std_logic;
          data: in std_logic_vector (1 downto 0);
               q: out  std_logic);
end;
 architecture rtl of ex is
 begin
    q<=a when data=“00” else
          b when data=“11” else
          c;
end;
```

Structural style example:

- download those files from Model in a new folder on the desktop.
- Add all files to a new project in Modelsim
- And then compile all and simulate File1. By force the values of A, B and CIN the output should be like that



A	B	CIN	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



File1

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

ENTITY FULLADDER IS PORT(A         : in  STD_LOGIC;
                         B         : in  STD_LOGIC;
                         CARRY_IN  : in  STD_LOGIC;
                         SUM       : out STD_LOGIC;
                         CARRY     : out STD_LOGIC);
END FULLADDER;

ARCHITECTURE STRUCTURAL of FULLADDER is
COMPONENT ORGATE   PORT (X : in  STD_LOGIC;
                         Y : in  STD_LOGIC;
                         Z : out STD_LOGIC);
END COMPONENT;

COMPONENT HALFADDER PORT (U     : in  STD_LOGIC;
                          V     : in  STD_LOGIC;
                          SUM   : out STD_LOGIC;
                          CARRY : out STD_LOGIC);
END COMPONENT;

  SIGNAL  W_SUM    : STD_LOGIC;
  SIGNAL  W_CARRY1 : STD_LOGIC;
  SIGNAL  W_CARRY2 : STD_LOGIC;
BEGIN
MODULE1: HALFADDER PORT MAP (A,B,W_SUM,W_CARRY1);
MODULE2: HALFADDER PORT MAP (W_SUM,CARRY_IN,SUM,W_CARRY2);
MODULE3: ORGATE    PORT MAP (W_CARRY1,W_CARRY2,CARRY);
END STRUCTURAL;
```

File2

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

ENTITY HALFADDER IS
 PORT(U,V:          IN  STD_LOGIC;
      SUM, CARRY : OUT STD_LOGIC);
END HALFADDER;

ARCHITECTURE RTL_HALFADDER OF HALFADDER IS
BEGIN
  SUM   <= U XOR V;
  CARRY <= U AND V;
END;
```

File3

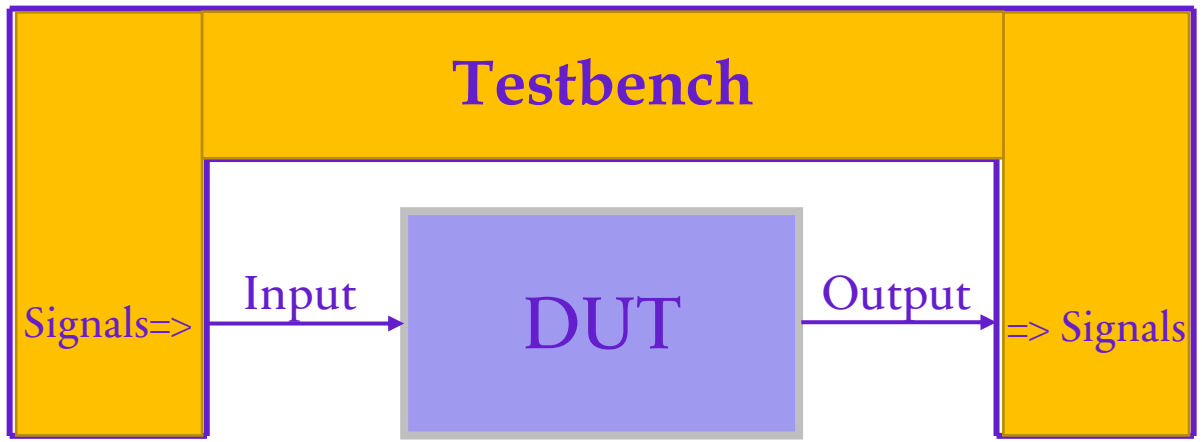
```
Ln#
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ORGATE IS
 PORT(X,Y : IN  STD_LOGIC;
          Z : OUT STD_LOGIC);
END ORGATE;

ARCHITECTURE RTL_ORGATE OF ORGATE IS
BEGIN
  Z <= X OR Y;
END RTL_ORGATE;
```

Testbench

- Testbench is an important part of VHDL design to check the functionality of Design through simulation waveform.
- Testbench provides a stimulus for **design under test** DUT or **Unit Under Test** UUT to check the output result.

- **A Test Bench consists of:**
  - Entity
    - has no ports (empty entity header)
  - Architecture
    - declares, instantiates, and wires together the driver model and the model under test
    - driver model provides the stimulus and verifies model responses



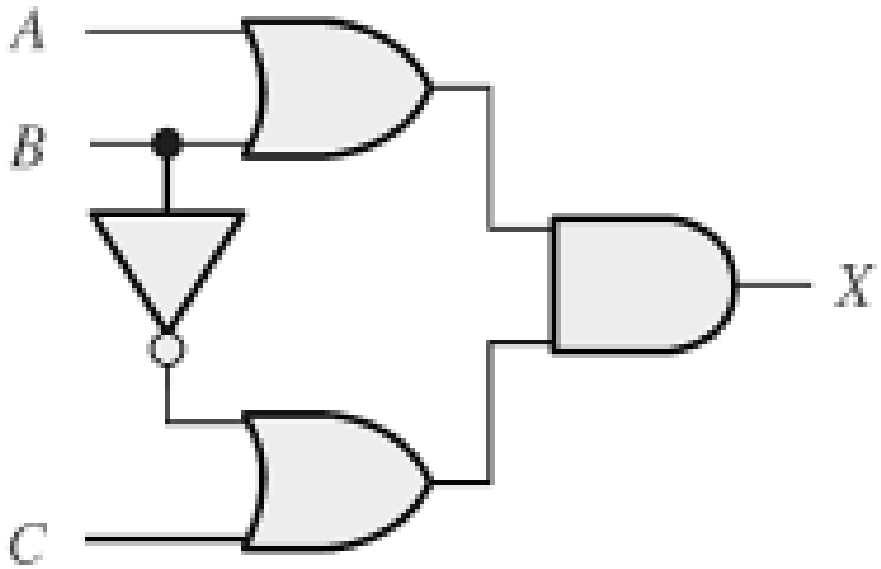
Full-Adder Testbench

```
Ln#
  1   LIBRARY IEEE;
  2   USE IEEE.STD_LOGIC_1164.ALL;
  3
  4   ENTITY TEST_FULLADDER IS
  5   END TEST_FULLADDER;
  6
  7   ARCHITECTURE TEST_BEHAVIORAL of TEST_FULLADDER is
  8
  9             SIGNAL A : STD_LOGIC;
 10             SIGNAL B : STD_LOGIC;
 11             SIGNAL CIN : STD_LOGIC;
 12             SIGNAL SUM : STD_LOGIC;
 13             SIGNAL CARRY : STD_LOGIC;
 14
 15             COMPONENT FULLADDER
 16                       PORT (A          : in STD_LOGIC;
 17                             B          : in STD_LOGIC;
 18                             CARRY_IN   : in STD_LOGIC;
 19                             SUM        : out STD_LOGIC;
 20                             CARRY      : out STD_LOGIC);
 21             END COMPONENT;
 22
 23   BEGIN
 24             DUT:FULLADDER
 25                       PORT MAP (A,B,CIN,SUM,CARRY);
 26
 27             STIMULUS: PROCESS
 28             BEGIN
 29                       A <= '0' ; B <= '0'; CIN <= '0';
 30                       WAIT FOR 100 NS ;
 31                       A <= '0' ; B <= '0'; CIN <= '1';
 32                       WAIT FOR 100 NS ;
 33                       .
 34                       .
 35                       .
 36             WAIT;
 37             END PROCESS;
 38   END TEST_BEHAVIORAL;
```

Group 1

- Write a VHDL Code to Implement the below circuit function using hierarchical modelling and data flow style for the components

- Use Signal for the internal connection between gates
- Verify your design by simulation, use Testbench to force different input patterns and check the output

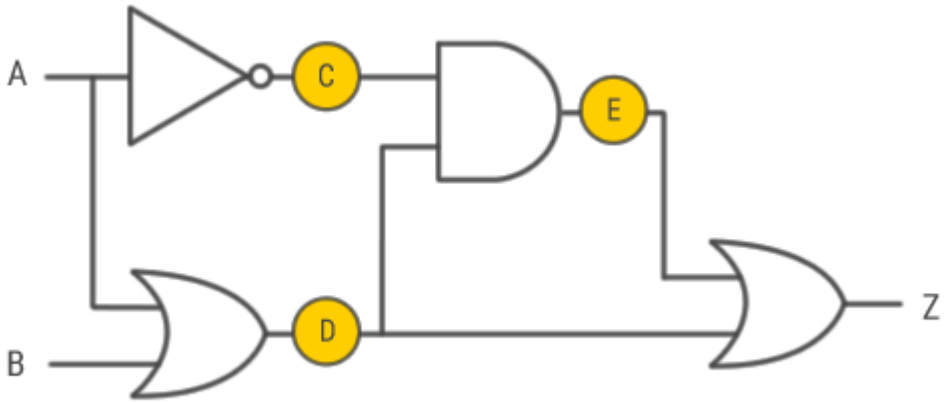


H.W:

Instead of a data flow style use a behavioral style utilizing the concurrent statements.

Group 2

- Write a VHDL Code to Implement the below circuit function using hierarchical modelling and data flow style for the components
- Use Signal for the internal connection between gates
- Verify your design by simulation, use Testbench to force different input patterns and check the output



H.W:
Instead of a data flow style use a behavioral style utilizing the concurrent statements.