Ho Chi Minh City University of Technology **LOGIC DESGIN WITH HDL**

Faculty of Computer Science and Engineering Practical session - Semester 222

WEEK 1

## Introduction &

**Structural Model**

# Introduction

## Aims

* + - Get familiar with Vivado software and the FPGA development flow.
    - Get familiar with FPGA Arty-Z7 board.
    - Practice in designing simple digital logic circuits with Verilog.
    - Understand the hierarchical design principle.
    - Practice in writing test benches for a designed module.

## Preparation

* + - Read the laboratory materials before class.
    - Revise chapter 0-3 about Verilog basic.
    - Each group prepare at least one laptop with Vivado software installed.

## Documents and lab materials

* + - M. Morris Mano, Michael D. Ciletti, Digital System with an Introduction to the Verilog HDL, VHDL, and SystemVerilog, Pearson Education, Inc, 2017
    - Lecture slides
    - *Arty-Z7-20-Master.xdc*: Arty-Z7 constraint file.
    - *Guide for Installing Vivado.pdf* : Guide for installing Vivado and getting started with Vivado and Arty-Z7.
    - *dec1to2.v*, *mux2to1.v* : 2-to-1 multiplexer module and its sub-module.
    - *mux2to2 tb.v* : test bench to simulate the module mux2to1.

## Procedure

For each exercise (also for further labs):

* + - Read the requirements, then determine the input/output signals of your circuits.
    - Make design idea of the circuit then using Verilog to model the circuit.
    - Analysis & Synthesis the circuit with Vivado software.
    - Write test bench to simulate the circuit on Vivado Simulator.
    - Generate the bitstream and program the Arty-Z7 to evaluate the circuit.

## Report requirements

* + - Lab exercises will be reviewed directly in class.
    - Write report (with circuit/simulation screenshots inserted) in pdf.
    - Must have group ID, group member’s names and student IDs in the report.
    - Compress the report with code files (only .v files) in only one .zip file, name the .zip the group ID (for example: Group1.zip).
    - Submit on BK-elearning by deadline.

# Exercises

## Exercise 1

1. Design a *1-to-2 decoder* using structure model as the following circuit:

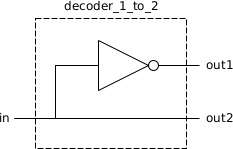
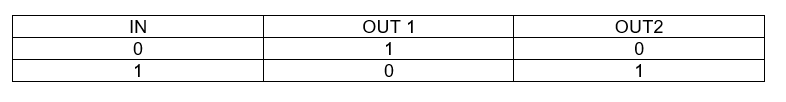


Figure 1: 1-to-2 decoder

* Dựa vào mạch ta thấy có một đầu vào (IN) và hai đầu ra (OUT 1, OUT2).
* Bảng chân trị



1. Design a *2-to-1 multiplexer* using structure model and hierarchical design (reuse the module decoder 1 to 2) as following circuit:

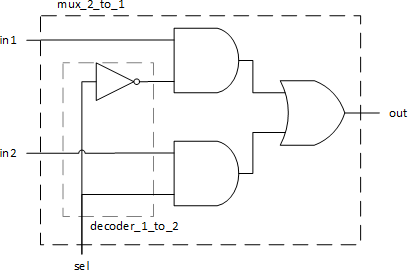


Figure 2: 2-to-1 multiplexer Check the schematic in RTL Analysis to see the result circuit.

module mux\_2\_to\_1(in1, in2, sel, out);

output out;

input in1, in2, sel;

wire T1, T2, Sbar;

and (T1, in2, sel), (T2, in1, Sbar);

not (Sbar, sel);

or (out, T1, T2);

endmodule

## Exercise 2

1. Write a test bench for the *2-to-1 Multiplexer* in *Exercise 1* then use Vivado Simulator to simulate the design, students can use the given example source code. Let’s analyse the structure of a test bench then point out the differences between an RTL code and a test bench code.

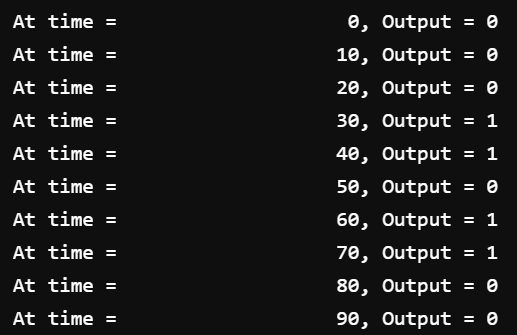
Change the **Radix**, **Format** of signals and use zoom tool to evaluate the waveform. Check the **Tcl console** window to see output of **$monitor** command.

1. Then, perform the Synthesis, compare the Synthesis’s Schematic and the RTL Analysis’s schematic.
2. After that, run the Implementation, check the Utilization report in Project Summary for used resources.
3. Add the Arty-Z7 constraint file to the project, assign pin for the design as follow:
   * + in1: btn[0], in2: btn[1]
     + sel: sw[0]
     + out: led[0]

then, generate bitstream file and program the FPGA to test the implemented circuit on board.

Graphical user interface, text, application

Description automatically generated



## Exercise 3

1. Design a half-adder circuit using structural model.

**a. Half-adder**

Text

Description automatically generated

**testbench:**

Text

Description automatically generated

**Text

Description automatically generated**

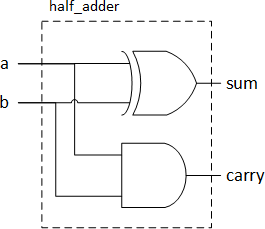


Figure 3: Half adder

1. Design a full-adder circuit using structural model. Reuse the half-adder module.

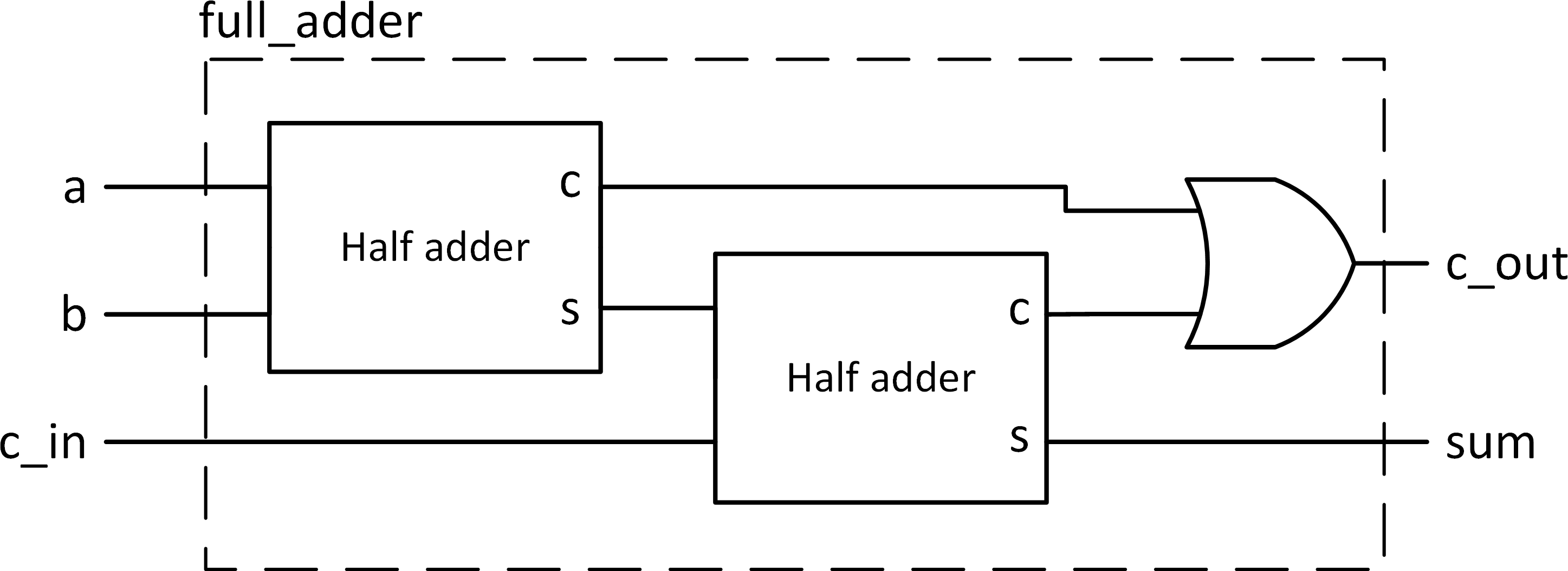


Figure 4: Full adder

Write a test bench to evaluate operations of the implemented full-adder circuit. Test the implemented circuit on Arty-Z7 board using switches/buttons and LEDs. *Bonus:* show the operands and sum on 7-seg LEDs.

**b.Full adder**

Text, letter

Description automatically generated

**testbench**

Graphical user interface, text, application

Description automatically generated

**Text

Description automatically generated**

1. Design a 4-bit ripple carry adder using structural model. Reuse the implemented full-adder. Write a test bench to simulate the implemented circuit.

A3 B3

A2 B2

A1 B1

A0 B0

C3 C2 C1 C0

FA

FA

FA

FA

C4 S3 S2 S1 S0

Figure 5: 4-bit ripple carry adder

**c. 4-bit ripple carry adder**

Text

Description automatically generated

**Testbench**

Graphical user interface, text, application

Description automatically generated

****

## Exercise 4

Give the 2-bit comparator circuit as Figure [6](#_bookmark1) with *A* = *{A*1*, A*0*}* and *B* = *{B*1*, B*0*}* are 2 2-bit input numbers, A gt B is active if *A > B*, A lt B is active if *A < B* and A eq B is active if *A* = *B*.

A1 A\_gt\_B



B1

A0 A\_lt\_B

B0

A\_eq\_B

Figure 6: 2-bit comparator Let’s design a 4-bit comparator following below steps:

* Analyse the functions of each output of the 2-bit comparator, then determine the functions of 4-bit

comparator outputs.

* Mạch so sánh 2 bit được hiển thị trong hình 6 so sánh hai số đầu vào 2 bit, là A và B. Các đầu ra của bộ so sánh như sau:

+ A\_eq\_B: Đầu ra này hoạt động khi và chỉ khi A=B.

+ A\_qt\_B: Đầu ra này hoạt động khi và chỉ khi A>B.

+ A\_lt\_B: Đầu ra này hoạt động khi và chỉ khi A<B.

* Các chức năng của đầu ra so sánh 4 bit có thể được xác định bằng cách mở rộng cùng một logic được sử dụng trong bộ so sánh 2 bit. Chúng ta có thể xem xét 4 bit đầu vào của A và B một cách riêng biệt, cụ thể như sau:

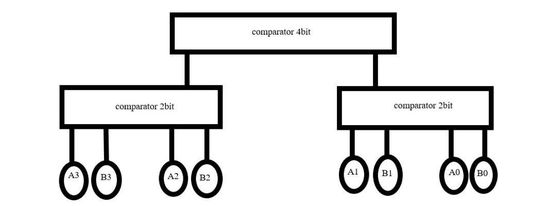
+ A\_eq\_B: Đầu ra này hoạt động khi và chỉ khi A=B. Trong một bộ sao sánh 4 bit, điều này có nghĩa là cả 4 bit đầu vào của A và B phải bằng nhau.

+ A\_gt\_B: Đầu ra này hoạt động khi và chỉ khi A>B. Trong một bộ so sánh 4 bit, điều này có nghĩa là A lớn hơn B theo bit quan trọng nhất (MSB) hoặc MSB bằng nhau và các bit quan trọng nhất tiếp theo của A lớn hơn các bit tương ứng của B.

* Conceptualize the design of 4-bit comparator from 2-bit comparators (the 2-bit comparator can be partitioned into smaller blocks). Draw a block diagram that describes the idea.
* Ý tưởng của bài toán so sánh này 4 bit là chúng ta sẽ chia thành 2 bit lớn và 2 bit nhỏ rồi đem chúng so sánh, rồi ta sử dụng các cổng luận lý gồm cổng and, cổng or để tổng hợp kết quả từ hai bộ sao sánh 2 bit, ta có thể tận dụng mạch so sánh 2 bit có sẵn trong bài toán này.
* Sơ đồ khối cho thiết kế này có thể được hiển thị như sau:



* Biểu đồ khối này bao gồm 2 bộ phận so sánh 2 bit. So sánh 2 bit lớn nhất của A và B và sao sánh 2 bit nhỏ nhất của A và B.
* Các đầu ra của hai bộ so sánh 2 bit sau đó được tổng hợp lại bằng các cổng luận lý gồm cổng and, cổng or và cho ra các kết quả.
* Draw a diagram to shows the designed circuit hierarchy.
* Dưới đây là một sơ đồ phân cách mạch ví dụ cho thấy thiết kế của bộ so sánh 4 bit sử dụng các bộ so sánh 2 bit làm khối xây dựng. Biểu đồ này cho thấy cấu trúc của bộ so sánh ở mức cao và không bao gồm các chi tiết về mạch bên trong của các bộ so sánh 2 bit.



* Trong sơ đồ này, bộ so sánh 4 bit được hiển thị ở cấp cao nhất, với 2 bộ so sánh 2 bit. Các bộ so sánh 2 bit này được sử dụng để só sánh 4 cặp bit tương ứng trong các số đầu vào 4 bit.
* Các bộ so sánh 2 bit là các khối xây dựng của thiết kế này và được tổ chức theo thứ bậc trong bộ so sánh. Các đầu vào cho mỗi bộ so sánh 2 bit là 2 cặp bit tương ứng trong các số đầu vào. Các đầu ra của các bộ so sánh 2 bit được kết hợp để tạo ra đầu ra cuối cùng của bộ sao sánh 4 bit.
* Các cấu trức phân cấp này giúp quản lý sự phức tạp của thiết kế bộ so sánh, giúp việc thực hiện và gỡ lỗi dễ dàng hơn.
* Implement the designed circuit using Verilog HDL structural model.

module comparator\_2bit(  
input [1:0] A,

input [1:0] B,

output A \_gt\_B,

output A \_It\_B,

output A\_eq\_B  
);  
wire va1, va2, va3, xnol, xno2;  
and (va1, ~A[1], B[1]),  
(va2, ~A[1), ~A[0], B[0]),

(va3, ~A[0], B[1], B[0]),  
(A\_eq\_B, xno1, xno2);  
xnor (xno1, A[1], B[1]),

(xno2, A [0], B[0]);  
Or (A\_It\_B, va1, va2, va3);  
nor (A\_gt\_B, A\_It\_B, A \_eq\_B);  
endmodule  
module comparator\_4bit(  
input [3:0] A,

input [3:0] B,

output A\_gt\_B,

output A\_It\_B,

output A\_eq\_B  
);  
wire gt1, gt2, It1, It2, eq1, eq2, va1, va2;

* Write a test bench to simulate the implemented circuit.

'timescale 1ns / 1ps  
module comparator\_4bit\_tb();  
reg [3:0] A;  
reg [3:0] B;  
wire A\_gt\_B;  
wire A\_It\_B;  
wire A\_eg\_B;  
comparator\_4bit comp4bit(A, B, A\_gt\_B, A\_It\_B, A\_eg\_B);

initial $monitor("time %t: A=%d, B= %d, A \_gt\_B=%b, A\_It\_B=%b, A\_eg\_B=%b\n", $time, A, B,

A\_gt\_B, A\_It\_B, A\_eg\_B);  
initial begin  
 {A, B} = 8'b0000\_0001;  
#10 {A, B} = 8'b1001 1001;  
#10 {A, B} = 8'b1011\_1110;  
#10 {A, B} = 8'b1101\_0100;  
#10 {A, B} = 8'b1111 1110;  
#10 {A, B} = 8'b1011\_0111;  
#10 {A, B} = 8'b1001\_0010;  
#10 {A, B} = 8'b0010\_0100;  
#10 {A, B} = 8'b1011\_0110;  
#10 {A, B} = 8'b0110\_0110;  
#10 $stop;  
end  
endmodule