Lab 3 – Ripple Carry Adder

CS1050 Computer Organization and Digital Design

Dept. of Computer Science and Engineering, University of Moratuwa

Learning Outcomes

In this lab we will design a 4-bit Ripple Carry Adder (RCA). After completing the lab, you will be able to:

- design and develop a Half Adder, Full Adder, and a Ripple Carry Adder
- build more complex components using several basic components
- verify their functionality via simulation and on the development board

Introduction

The adder/subtractor unit is perhaps the most important element of an Arithmetic and Logic Unit (ALU) of a microprocessor. In this lab, we will also learn about hierarchical design where more complex components are built using many basic components. Those high-level components may be further combined to build even larger components. For example, we will use 2 Half Adders (Has) to build a Full Adder (FA) and multiple FAs to build the RCA. Later in the class and labs you will also learn how to extend your RCA to support subtraction.

Building the Circuits

We will first build an HA, then an FA, and finally the 4-bit RCA.

Step 1: Finding the Boolean expressions for HA and FA.

Write the truth table for a HA and FA. Simplify the HA and FA Boolean equations such that FA can be built using 2 HAs.

Step 2: Building HA.

Create a new project in Xilinx Vivado and name it as **Lab 3**. Build a HA using basic logic gates such as AND, OR, XOR, or NOT. Name your file as **HA**. Label the inputs to HA as **A** and **B** and outputs as **S** and **C**.

Test the functionality of the HA by simulating the circuit. Name the Test Bench File as **TB_HA**.

Step 3: Building FA.

To add a new source file to the project, click on the **+** button or press **Alt + A** in the **Sources** pane of the **Project Manager** panel. Name the file as **FA**.

Label the inputs to FA as **A**, **B**, and **C_in** and outputs as **S** and **C_out**.

To build a FA using HAs and basic logic gates, we need to fist add the already built HA as a component to our design. Therefore, add the following code after the line that says `architecture Behavioral of FA is`:

```
component HA
    port (
    A: in std_logic;
    B: in std_logic;
    S: out std_logic;
    C: out std_logic);
end component;
```

A `component` declaration is used to define the interface to the lower-level design entity HA. This essentially allow one entity (aka. module) to be used as part of another entity. The component declaration must be placed in the declaration section of the architecture body.

Then we need to initiate 2 HAs from this entity. Let us label our HAs as **HA0** and **HA1**. Add the following code after the `begin` keyword:

```
HA_0 : HA
    port map (
    A => A,
    B => B,
    S => HA0_S,
    C => HA0_C);

HA_1 : HA
    port map (
    A => HA0_S,
    B => C_in,
    S => HA1_S,
    C => HA1_C);
```

Above code initiates 2 instances of the entity HA and connects them to the inputs and outputs of the FA we are trying build.

`HA0_S` and `HA0_C` refer to the outputs of `HA0` while `HA1_S` and `HA2_C` refer to the outputs of `HA1`. These are internal signals; hence, need to be defined using signal keyword as follows:

```
SIGNAL HA0_S, HA0_C, HA1_S, HA1_C : std_logic;
```

Make sure the above line is added after declaring the component (i.e., after `end component;` line) and before the `begin` statement.

Then add the relevant VHDL code to define **S** and **C_out** of the FA. Now you have defined the behavior of the FA.

As our project now has a FA and HA we need to define which entity is the higher-level design (aka. top-level design) so that the Synthesis and Implementation know which circuit to build. In this case FA is the higher-level entity while HA is the lower-level entity, as FA is built using HAs.

Click on **FA(Behavioral)(FA.vhd)**. Right click and then select **Set as Top** from the popup menu. Once this is done you will see a small icon appearing before **FA(Behavioral)(FA.vhd)**.

Check the schematic view of the developed FA.

Test the functionality of the FA by simulating the circuit. Name the Test Bench File as **TB_FA**.

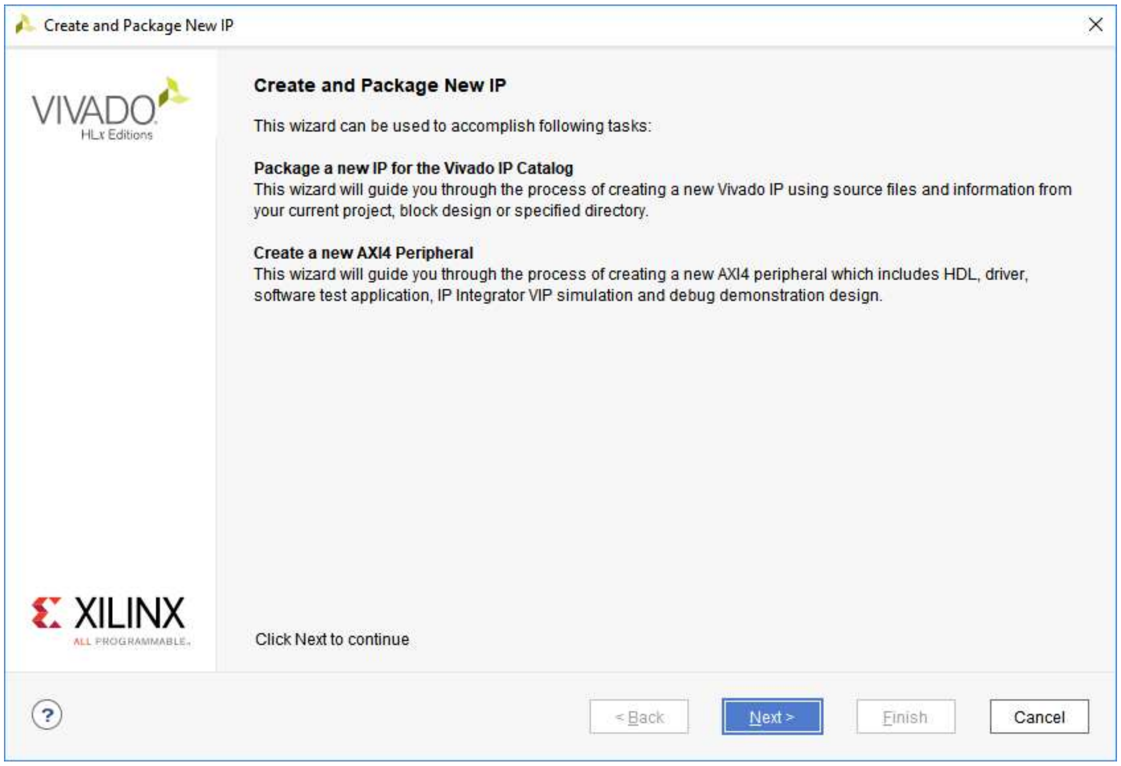
Step 4: Creating an FA symbol.

We will need FAs in future designs. Therefore, we can create a macro symbol (referred to as a Block in Vivado) for future use of this FA.

To create a macro circuit, select **FA(Behavioral)(FA.vhd)** under Design Sources and then select **Tools** → **Create and Package New IP** from the menu.

This will open the Create and Package new IP Wizard dialogue box (see Fig. 1).

Click **Next >**.


Create and Package New IP

Create and Package New IP

This wizard can be used to accomplish following tasks:

Package a new IP for the Vivado IP Catalog
This wizard will guide you through the process of creating a new Vivado IP using source files and information from your current project, block design or specified directory.

Create a new AXI4 Peripheral
This wizard will guide you through the process of creating a new AXI4 peripheral which includes HDL, driver, software test application, IP Integrator VIP simulation and debug demonstration design.

Click Next to continue

Figure 1 – Create and package new IP window.

In electronic design a semiconductor Intellectual Property (IP) core, IP core, or IP block is a reusable unit of logic, cell, or chip layout design that is the intellectual property of one party and can be used by others in the designs.

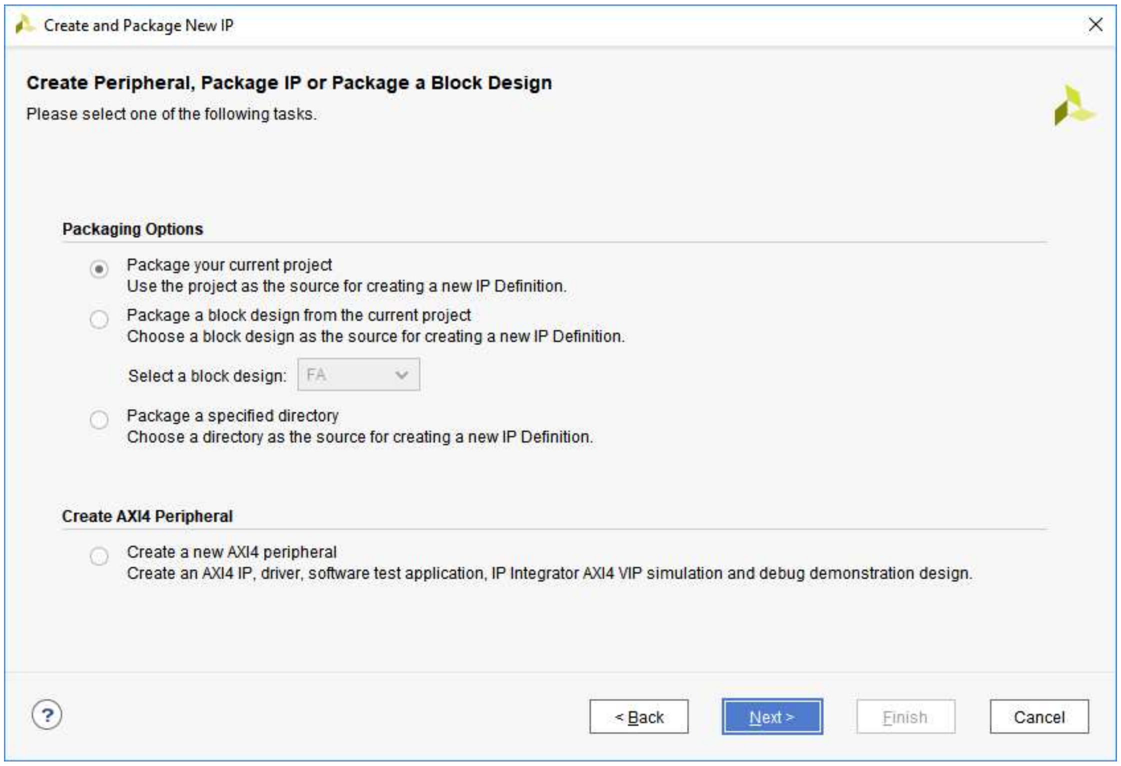
Select **Package your current project** (most probably this will be selected by default as seen in Fig. 2). Click **Next >**.

Then in the next window set the **IP location** to where you want to save the files (see Fig. 3). Make sure the file path is a location that is accessible to you such that you can save the files for later use. Click **Next >**.

Click **Finish** to close the window.

This opens the Vivado IP packager GUI (see Fig. 4), where you can view sources, IP name, etc. You name set your name as **Vendor:**.

Scroll down to the end of packaging steps, and select **Review and Package.** Select **Package IP.** You may delete the temporary project and close the window. Now your FA can be used as an IP block in other new Vivado projects. In a new Vivado project, you can click on **IP Catalog** under **Project Manager** (In Flow Navigator Pane).



Create Peripheral, Package IP or Package a Block Design

Please select one of the following tasks.

Packaging Options

[✓ Package your current project]
Use the project as the source for creating a new IP Definition.

Package a block design from the current project
Choose a block design as the source for creating a new IP Definition.

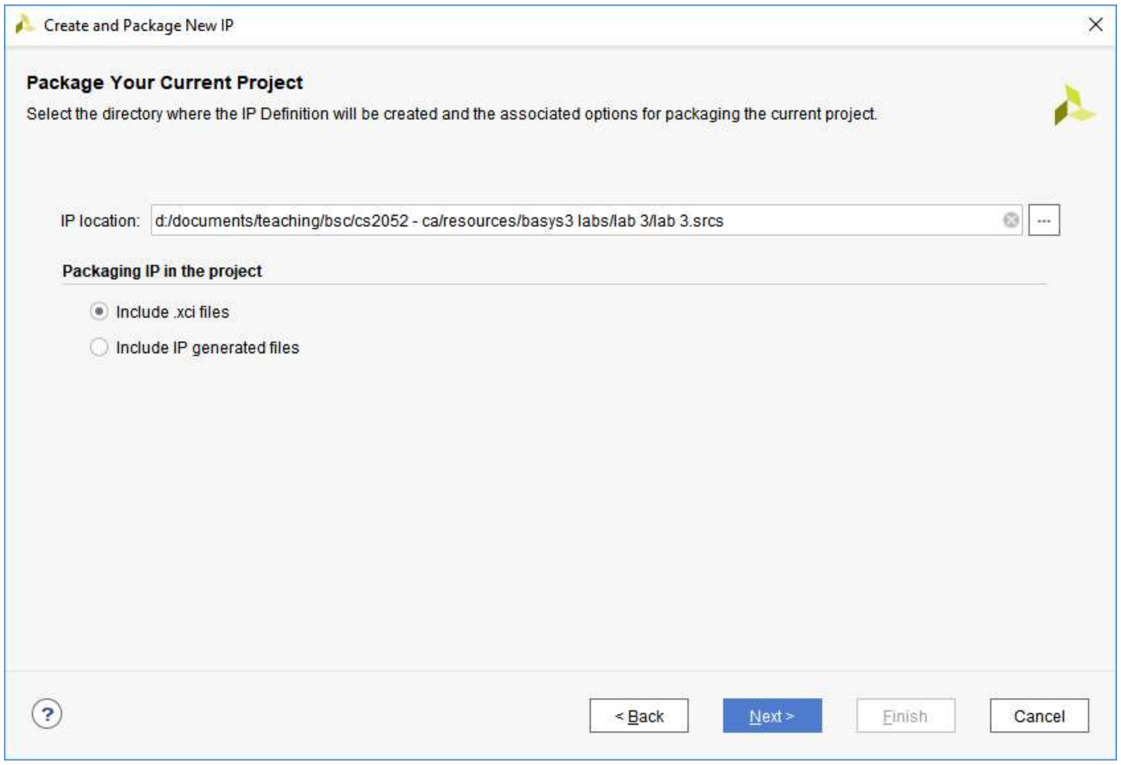
Select a block design: FA

Package a specified directory
Choose a directory as the source for creating a new IP Definition.

Create AXI4 Peripheral

Create a new AXI4 peripheral
Create an AXI4 IP, driver, software test application, IP Integrator AXI4 VIP simulation and debug demonstration design.

Figure 2 – Package options window.



Package Your Current Project

Select the directory where the IP Definition will be created and the associated options for packaging the current project.

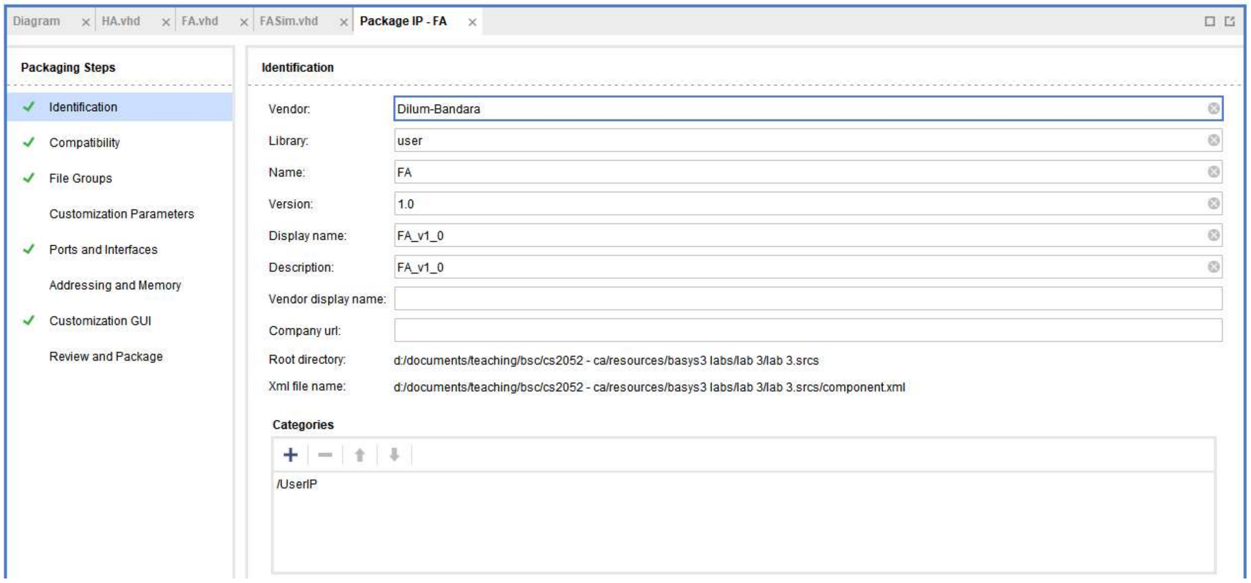
IP location: d:/documents/teaching/bsc/cs2052 - ca/resources/basys3 labs/lab 3/lab 3.srcs

Packaging IP in the project

[✓ Include .xci files]

Include IP generated files

Figure 3 – IP path options window.



Packaging Steps

- ✓ Identification
- ✓ Compatibility
- ✓ File Groups
- Customization Parameters
- ✓ Ports and Interfaces
- Addressing and Memory
- ✓ Customization GUI
- Review and Package

Identification

Vendor:	Dilum-Bandara
Library:	user
Name:	FA
Version:	1.0
Display name:	FA_v1_0
Description:	FA_v1_0
Vendor display name:	
Company url:	
Root directory:	d:/documents/teaching/bsc/cs2052 - ca/resources/basys3 labs/lab 3/lab 3.srcs
Xml file name:	d:/documents/teaching/bsc/cs2052 - ca/resources/basys3 labs/lab 3/lab 3.srcs/component.xml

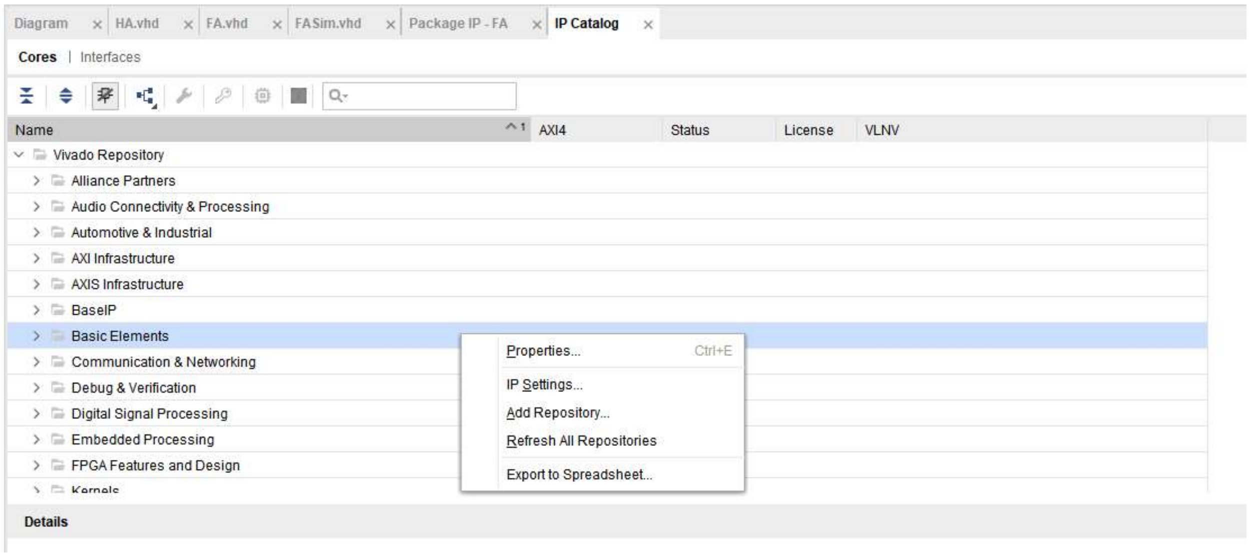
Categories

/UserIP

Figure 4 – Vivado IP packager view.

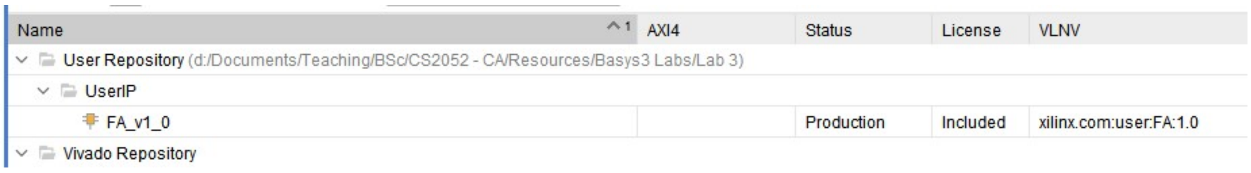
This opens the IP Catalog. Right click anywhere on the catalog and select **Add Repository...** from the popup menu (see Fig. 5).

Browse to the file path where you save the IP files. Then click **Select** button. As seen in Fig. 6 now your IP should get listed under UserIP. This step is not essential in this lab. But it will be required in future labs.



Name	AXI4	Status	License	VLNV
Vivado Repository				
Alliance Partners				
Audio Connectivity & Processing				
Automotive & Industrial				
AXI Infrastructure				
AXIS Infrastructure				
BaseIP				
Basic Elements				
Communication & Networking				
Debug & Verification				
Digital Signal Processing				
Embedded Processing				
FPGA Features and Design				
Kernels				

Figure 5 – IP Catalog view.



Name	AXI4	Status	License	VLNV
User Repository (d:/Documents/Teaching/BSc/CS2052 - CA/Resources/Basys3 Labs/Lab 3)				
UserIP				
FA_v1_0		Production	Included	xilinx.com:user:FA:1.0
Vivado Repository				

Figure 6 – UserIP listed in IP Catalog.

Step 5: Build a 4-bit RCA.

Create a new schematic file and save it as **RCA_4**. Now build a 4-bit RCA using multiple FAs. Label the inputs to RCA as **A0-A3**, **B0-B3**, and **C_in**. Label the outputs as **S0-S3** and **C_out**.

Create 4 instances of the FA as **FA0**, **FA1**, **FA2**, and **FA3**. Connect the inputs and outputs as required. Make sure to set **C_in** of **FA0** to ground. Your final VHDL code should look similar to the following:

```
entity RCA_4 is
    Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           A3 : in STD_LOGIC;
           B0 : in STD_LOGIC;
           B1 : in STD_LOGIC;
           B2 : in STD_LOGIC;
           B3 : in STD_LOGIC;
           C_in : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           S3 : out STD_LOGIC;
           C_out : out STD_LOGIC);
end RCA_4;

architecture Behavioral of RCA_4 is

    component FA
        port (
        A: in std_logic;
        B: in std_logic;
        C_in: in std_logic;
        S: out std_logic;
        C_out: out std_logic);
    end component;

    SIGNAL FA0_S, FA0_C, FA1_S, FA1_C, FA2_S, FA2_C, FA3_S, FA3_C
      : std_logic;

begin
    FA_0 : FA
        port map (
            A => A0,
            B => B0,
            C_in => '0', -- Set to ground    C_in => C_in,
            S => S0,
            C_Out => FA0_C);

    FA_1 : FA
        port map (
            A => A1,
            B => B1,
            C_in => FA0_C,
            S => S1,
            C_Out => FA1_C);
    FA_2 : FA
        port map (
            A => A2,
            B => B2,
            C_in => FA1_C,
            S => S2,
            C_Out => FA2_C);

    FA_3 : FA
        port map (
            A => A3,
            B => B3,
            C_in => FA2_C,
            S => S3,
            C_Out => C_out);
end Behavioral;
```

Now we need to set RCA as the top-level entity, as it is built using multiple FAs. For this right click on **RCA_4(Behavioral)(RCA_4.vhd) and** then select **Set as Top** from the popup menu. This is also essential as input and output pins from BASYS 3 are connected only to the top-level design which is the RCA.

Simulate the RCA and make sure it functions correctly. Name the Test Bench File as **TB_4_RCA**. It is not essential to try all the possible combinations of inputs. Instead, you should at least try the following input combinations:

- Consider the 6 digits of your index number. Then convert your index number to binary. Then take 4 Least Significant Bits (LSBs) and add them with the next 4 LSBs. Then try the next set of 4-bits (ignore any remaining bits). For example, suppose your index number is 123456R. Then its binary representation is 01 1110 0010 0100 0000 (ignoring the check digit). Then try 0000 + 0100 and 0010 + 1110.
- 0101 + 1011 and 0111 + 1111
- Any 4 other unique combinations

Step 6: Connecting inputs and outputs.

Connect switches SW0-SW3 as the inputs **A0-A3** and SW12-SW15 as the inputs **B0-B3**. Connect outputs **S0-S3** to LED LD0-LD3 and **C_out** to LED LD15.

Check the schematic view.

Step 7: Test on BASYS3.

Generate the programming file (i.e., bitstream) and load it to the BASYS 3 board.

Change the switches on the board and verify the functionality of your RCA (check the output of LEDs).

Demonstrate the circuit to the instructor and get the Lab Completion Log singed.

Step 8: Lab Report

You need to submit a report for this lab. Your report should include the following:

- Student name, index number, and group. Do not attach a separate front page
- State the assigned lab task in a few sentences

- Truth tables and steps involved in simplifying the Boolean expressions
- All VHDL files
- All timing diagrams. Show all possible inputs for HA and FA. For 4-bit RCA provide at least the inputs listed under Step 5
- Discuss why some of the input combinations results in outputs that cannot be represented using LED LD0-LD3. Discuss the role of LD15
- Other conclusions from the lab

Submit the lab report to the moodle before the mentioned deadline.

Prepared By

- Dilum Bandara, PhD – Feb 26, 2014
- Updated on Sep 27, 2018

  Updated May 25, 2022