EE2026: DIGITAL SYSTEMS

Academic Year 2017-2018, Semester 2

LAB 2: Combinational Circuits in Verilog

OVERVIEW:

A combinational circuit is one where the outputs depend only on the current inputs. In this lab, we will be designing some combinational circuits that are able to perform addition.

The pre-requisite for this lab requires one to be able to:

- Create a Verilog project and design source in Vivado.
- Create a testbench to simulate the design source.
- Generate the RTL schematic and the synthesised circuit schematic of design source.
- Map and implement a design on the Basys 3 development board.

This lab will cover the following:

- Designing a one-bit full adder circuit using the dataflow modelling method.
- Designing a two-bit parallel adder circuit using the structural modelling method.

Tasks for this lab include:

- Designing the Verilog code of a one-bit full adder.
- Designing, simulating and implementing a four-bit parallel adder on the FPGA.
- Designing and simulating a four-bit subtractor.

ONE-BIT-FULL ADDER:

Consider the binary addition shown in ***Figure 2.1***. To design a circuit that would perform the addition of two one-bit values, the circuit would need to have three input bits and two output bits.

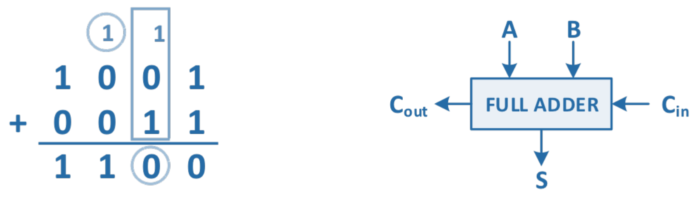


Figure 2.1: *Binary Addition and functional block diagram of the one-bit full adder*

Such a circuit is called a one-bit full adder. It adds two bits (**A**, **B**) and the carry (**C_{in}**) from a previous stage of addition, and produces a sum (**S**) and a carry (**C_{out}**), as illustrated through a truth table in ***Figure 2.2***. By simplifying the truth table, the Boolean expressions for **S** and **C_{out}** can be obtained.

A	B	C_{in}	S	C_{out}
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

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + C_{in}\,(A \oplus B)$$

Figure 2.2: *Truth table and boolean expressions of the one-bit full adder*

Note: A half adder, in contrast to a full adder, does not involve a carry input. Thus, for a half adder, $S = A \oplus B$, and $C_{out} = AB$

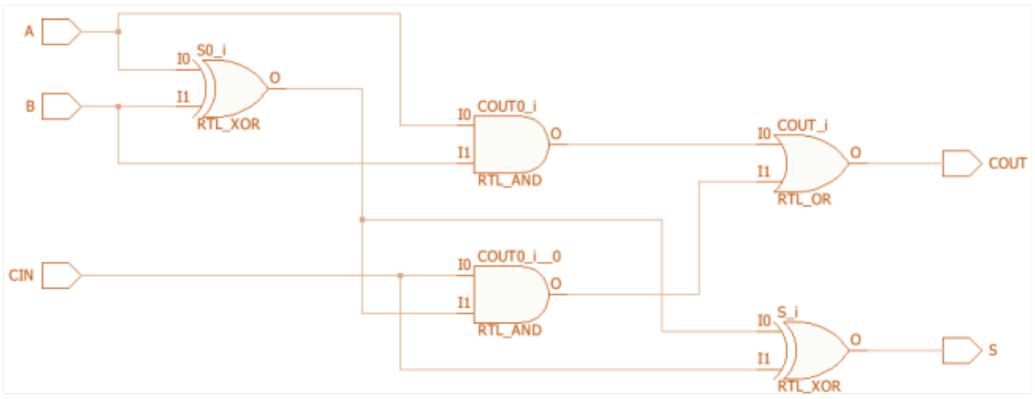
UNDERSTANDING | TASK 1

Using the dataflow method, complete the Verilog code for the one-bit full adder. Verify that the RTL schematic is as shown.

Verilog skeleton code for the one-bit-full adder:

```
module my_full_adder(input A, B, CIN, output S, COUT);

    assign S =
    assign COUT =

endmodule
```

RTL schematic for the one-bit full adder:



TWO-BIT FULL ADDER:

By cascading one-bit full adder blocks, the one-bit adder can be reused and parallel adders that add multiple bits can be created through the structural modelling method. A two-bit ripple-carry adder is illustrated in ***Figure 2.3***.

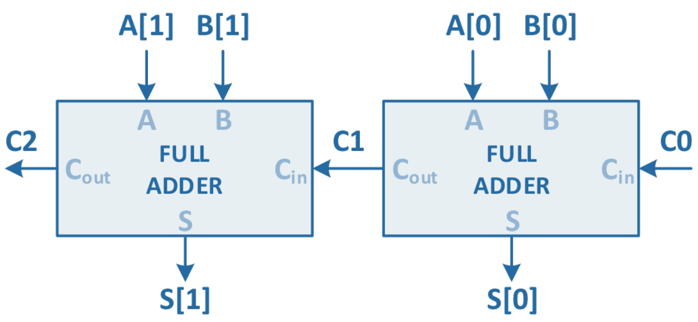


Figure 2.3: *Functional block diagram of the two-bit ripple-carry adder*

With the code for a one-bit full adder, a new module is created and two full adder blocks (fa0, fa1) are instantiated. By specifying the inputs and outputs to these blocks, the connection **C1** between them is made. Note that the order of signals during instantiation should respect the order in which they were declared in the one-bit full adder module **my_full_adder**.

This approach to hardware description is called structural modelling, whereby a more abstract module (for example, **my_2_bit_adder**) is built from simpler components describing gate-level hardware (such as **my_full_adder**).

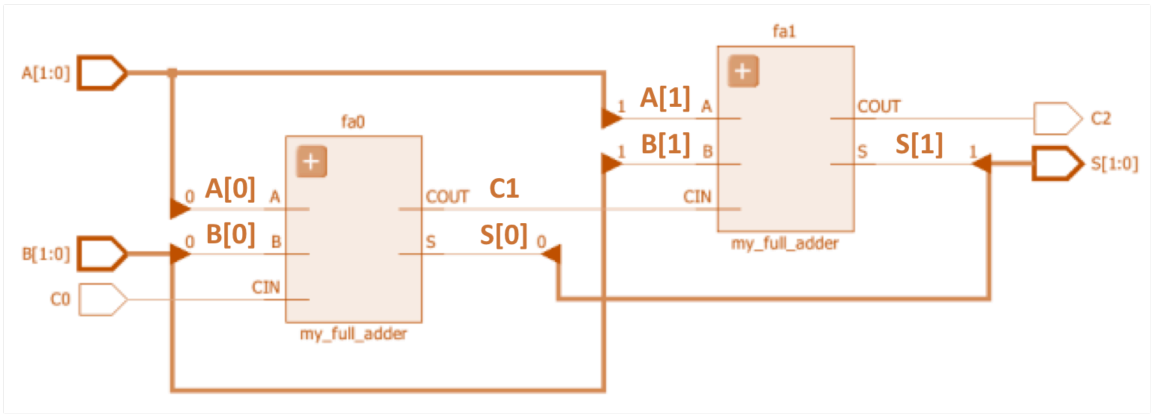
Verilog code for two-bit ripple-carry adder, using structural modelling:

```
module my_2_bit_adder(input [1:0] A, input [1:0] B, input C0,
                      output [1:0] S, output C2);

    wire C1;

    my_full_adder fa0 (A[0], B[0], C0, S[0], C1);
    my_full_adder fa1 (A[1], B[1], C1, S[1], C2);

endmodule
```

[Note] Two-bit port declaration for A, B, S:

Instead of having multiple input and output ports (A1, A0, B1, B0, S1, S0), multi-bit vector ports are defined.
`input [5:0] apple` means:
Input port named **apple**, with size of 6 bits. apple[5] refers to the MSB, apple[0] refers to the LSB.

RTL schematic for the two-bit ripple-carry adder:



FOUR-BIT FULL ADDER:

The functional block diagram of a four-bit ripple-carry adder is shown in ***Figure 2.4***.

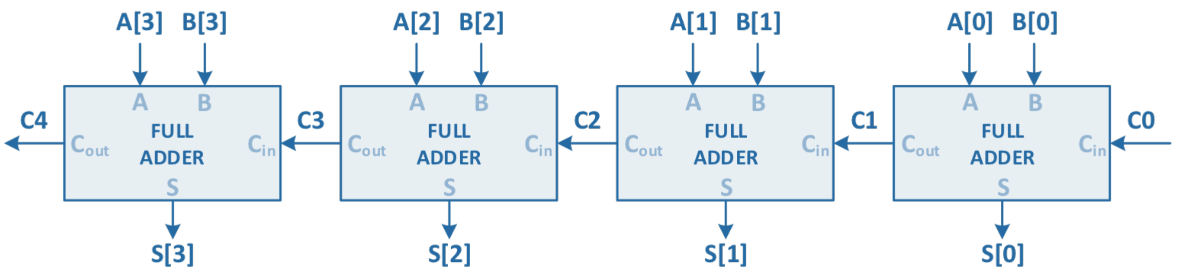


Figure 2.4: *Functional block diagram of a four-bit ripple-carry adder*

UNDERSTANDING | TASK 2

Start by adding a new design source for the four-bit full adder.

1. By using the structural modelling method, design a four-bit ripple-carry adder.
2. Generate the RTL schematic and check that connections between the blocks are correct.
3. Simulate your code with the following three sets of input values, and check that the simulation outputs are correct:

$$\begin{array}{lr} \text{A:} & 0\ \ 0\ \ 1\ \ 1 \\ \text{B:}\ + & \underline{0\ \ 0\ \ 1\ \ 1} \\ & \underline{\square\ \square\ \square\ \square} \end{array} \qquad \begin{array}{lr} \text{A:} & 1\ \ 0\ \ 1\ \ 1 \\ \text{B:}\ + & \underline{0\ \ 1\ \ 1\ \ 1} \\ & \underline{\square\ \square\ \square\ \square} \end{array} \qquad \begin{array}{lr} \text{A:} & 1\ \ 1\ \ 1\ \ 1 \\ \text{B:}\ + & \underline{1\ \ 1\ \ 1\ \ 1} \\ & \underline{\square\ \square\ \square\ \square} \end{array}$$

[Note] Brief guidelines for multi-bit vector ports in the testbench:

▶ The port size should be indicated when declaring the signals. Inputs to the *dut* are declared using **reg**, whereas outputs from the *dut* are declared using **wire**:
```
reg [3:0] A; wire [3:0] S;
```

▶ The parameters for the *dut* do not need the port size of the signals:
```
my_4_bit_adder dut (A, B, CARRY_IN, S, CARRY_OUT);
```

▶ The testbench stimuli for multi-bit vector ports can be written as:
```
A = 4'b0011; B = 4'b0011; CARRY_IN = 1'b0;
```

4. Synthesise and implement your code on the FPGA, using appropriate switches and LEDs on the Basys 3 development board to represent the inputs and outputs. Consider using the Xilinx's design constraint file template (**Basys3_Master.xdc**) for ease of implementation.

This task is considered completed and understood if you have the following items related to the four-bit full adder:

- The RTL schematic of your design (item 2 of this task)
- The simulation waveform results of the three testbench stimuli (item 3 of this task)
- The four-bit ripple-carry adder on the Basys 3 development board (item 4 of this task)

UNDERSTANDING | TASK 3

Instead of using four one-bit adder blocks, can you think of an alternative way (still using structural modelling) in creating the four-bit ripple-carry adder? Consider doing the same things as indicated in UNDERSTANDING | TASK 2 by using such alternative way. This task is meant as practice for you to improve your Vivado skills and Verilog understanding, and will not be explained.

POST-LAB TASK FOR LAB 2 – THE SUBTRACTOR:

The final task of this lab is to create a four-bit subtractor, by making use of the four-bit parallel adder that had been created in **UNDERSTANDING | TASK 2**

Since $A - B = A + (-B)$, by adding A to $-B$, the answer to $A - B$ can be obtained.
1's or 2's complement can be used to find $-B$ from B.

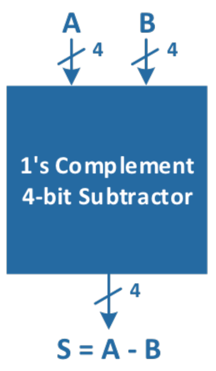
Briefly explain how one can design a complete four-bit subtractor circuit, and also include in the report the Verilog code to implement the circuit, **depending on the official lab session you are assigned to**:

- **Monday's EE2026 session** students must make compulsory use of **2's complement** for the four-bit subtractor.
- **Tuesday's EE2026 session** students must make compulsory use of **1's complement** for the four-bit subtractor.
- **Wednesday's EE2026 session** students must make compulsory use of **2's complement** for the four-bit subtractor.
- **Friday's EE2026 session** students must make compulsory use of **1's complement** for the four-bit subtractor.

All reports contents (explanation, Verilog codes, simulation codes) are to be completed within 1 sheet (2 pages)

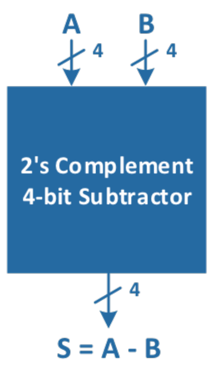
Examples on the expected output S, for some input values of A and B

One's Complement



A	B	S
0101 (+5)	0001 (+1)	0100 (+4)
0101 (+5)	1110 (-1)	0110 (+6)
1110 (-1)	0101 (+5)	1001 (-6)
1110 (-1)	1010 (-5)	0100 (+4)

Two's Complement



A	B	S
0101 (+5)	0001 (+1)	0100 (+4)
0101 (+5)	1111 (-1)	0110 (+6)
1111 (-1)	0101 (+5)	1010 (-6)
1111 (-1)	1011 (-5)	0100 (+4)

Further instructions:

- The addition and subtraction operators (**+**, **−**) are **not allowed** within the code
- As much as possible, make your code efficient and reduce redundancies.
- Improve the subtractor's ease-of-use by considering the user's point of view.

Include the following in the report:

- Name, official EE2026 lab session day, and matriculation number.
- A brief explanation of the design that will allow for a four-bit subtractor
- Verilog codes for all the modules used in the design, including the simulation codes
- Simulation waveform results with the **4 test cases** (Choose the set of test cases that corresponds to the complement method that you have used) **mentioned in** "**Examples on the expected output S, for some input values of A and B**". Ensure that they are all readable in the screenshot.
- G.A.s and T.A.s will not provide additional information on what can be included.
- You are not required to show the FPGA implementation on the Basys 3 development board.

Submission instructions:

- Maximum of two A4 pages (1 sheet).
- Font size can vary between 6 to 20, depending on how much information you wish to fit in.
- Report should be as a softcopy. Hardcopies are not accepted.
- Report in PDF format, with good quality screenshots (readable simulation waveforms, no camera images).
- Naming of the report to follow this template as close as possible:
  Lab2_Official lab session day_**Name as indicated on your matriculation card**_Matriculation number_Report
  Examples:
  Lab2_Monday_John Doe Jr._A0131086X_Report
  Lab2_Wednesday_Jane Junior_A0202026_Report
- Upload to IVLE, in the folder corresponding to your official lab session day.
- The IVLE upload must be completed within 6 days of your official lab session day. If you have attended other lab session days on a temporary basis, your official lab session day is the one considered for the upload deadline.

****

Feedbacks: (Note that feedbacks, whether positive or negative, DO NOT have any effects on your grades ☺)

To enhance your learning experience, please include changes and improvements you would like to have regarding the lab manuals, sessions, and guidance provided.