

EE2026: DIGITAL DESIGN

Academic Year 2021-2022, 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.
- Understand well the contents of Lectures 1, 2, and especially 3: Introduction to Verilog.

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.
- Understanding a 1-bit two-to-one multiplexer.

GRADED ASSIGNMENT [LUMINUS SUBMISSION: WEDNESDAY 9th FEBRUARY 2022, NOON]:

Further details are available at the end of this lab manual

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 $C_{out} = AB$

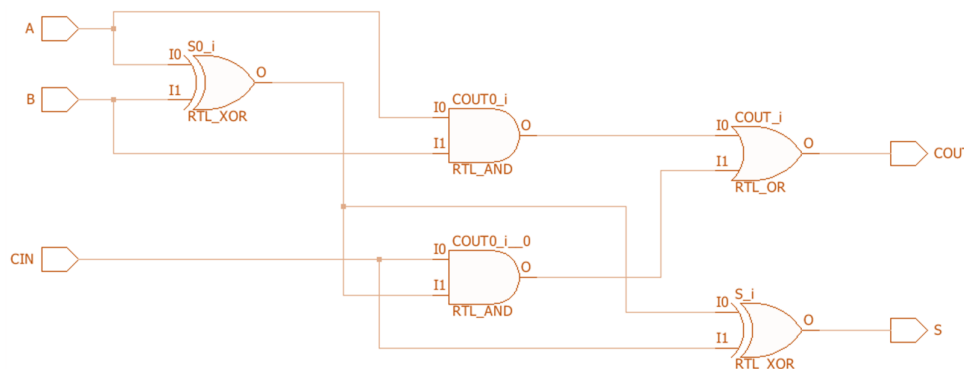
UNDERSTANDING | TASK 1

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*.

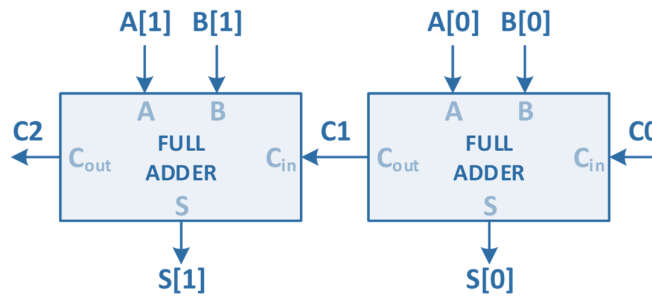


Figure 2.3: Functional block diagram of the two-bit ripple-carry adder

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
```

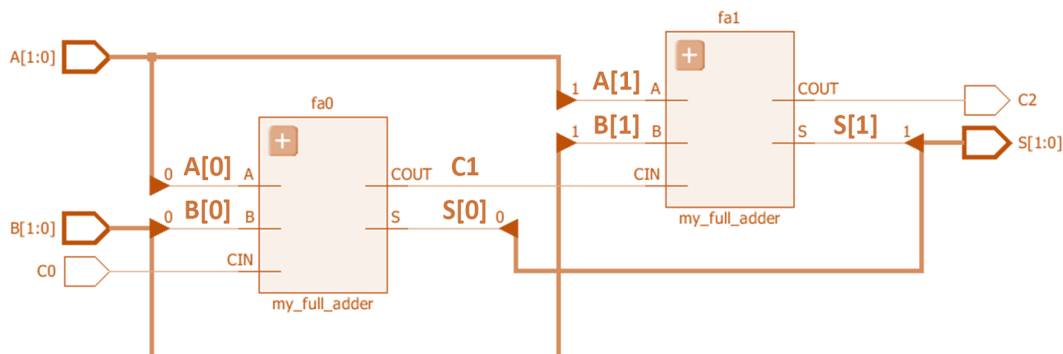
[Note] Two-bit port declaration for A, B, S:

Instead of having multiple input and output ports (A1, A0, B1, B0, S1, S0), multi-bit vector ports are defined.

Example: `input [5:0] apple`

Input port name is **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*.

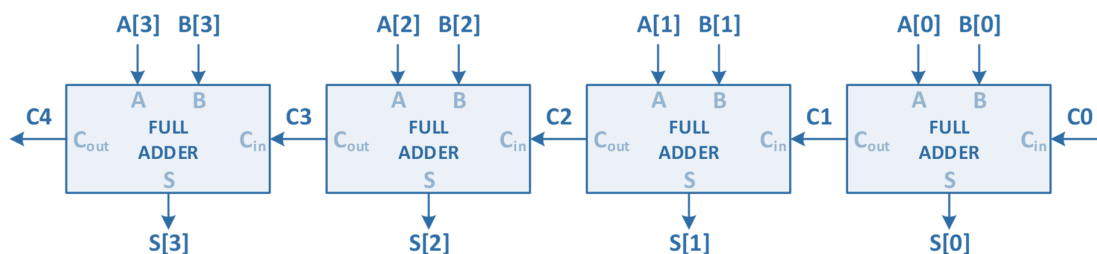


Figure 2.4: Functional block diagram of a four-bit ripple-carry adder

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:

$$\begin{array}{r} A: \quad 0 \ 0 \ 1 \ 1 \\ B: + \ 0 \ 0 \ 1 \ 1 \\ \hline \square \ \square \ \square \ \square \end{array}$$

$$\begin{array}{r} A: \quad 1 \ 0 \ 1 \ 1 \\ B: + \ 0 \ 1 \ 1 \ 1 \\ \hline \square \ \square \ \square \ \square \end{array}$$

$$\begin{array}{r} A: \quad 1 \ 1 \ 1 \ 1 \\ B: + \ 1 \ 1 \ 1 \ 1 \\ \hline \square \ \square \ \square \ \square \end{array}$$

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

- The port size should be indicated when declaring the signals. Inputs to the module being tested are declared using **reg**, whereas outputs from the module being tested are declared using **wire**:
`reg [3:0] A; wire [3:0] S;`
- The parameters for the module being tested do not need the port size of the signals:
`my_4_bit_adder module_alias (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.

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.

1-BIT TWO-TO-ONE MULTIPLEXER (POST-LAB NON-GRADED – NO SUBMISSION REQUIRED):

A multiplexer (MUX) is a combinational circuit that connects one of its input signals to the output, based on the control signal. A simple 1-bit two-to-one mux, with inputs **A**, **B**, control signal **S**, and output **Z**, is illustrated as a functional block diagram, together with its simplified truth table, in **Figure 2.5**.

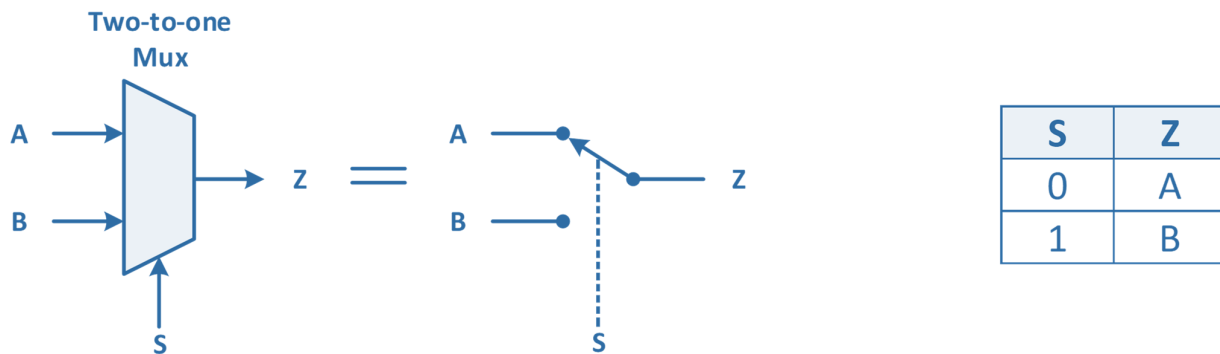


Figure 2.5: Functional block diagram and truth table of a 1-bit two-to-one multiplexer

UNDERSTANDING | TASK 4

A quick way to implement the Verilog code for a 1-bit two-to-one multiplexer is using the conditional syntax:

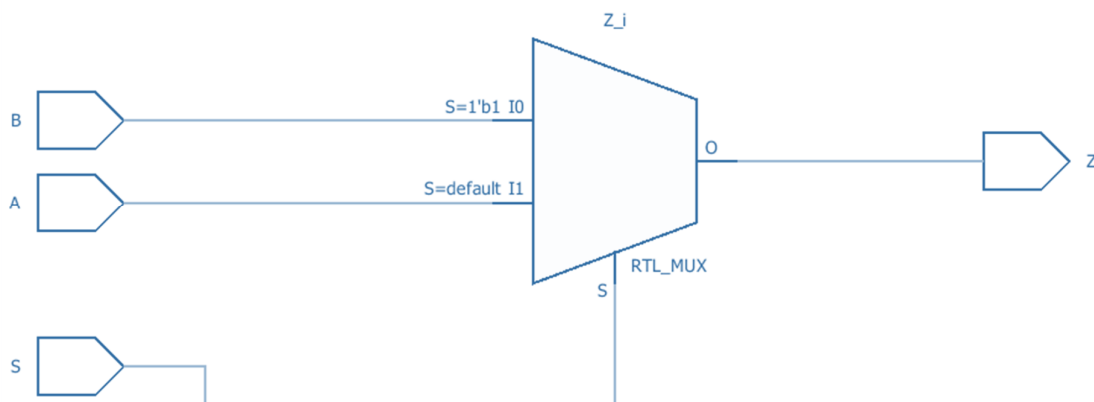
condition ? expression1 : expression2;

Notice in the schematic, how the code is automatically recognised as a MUX.

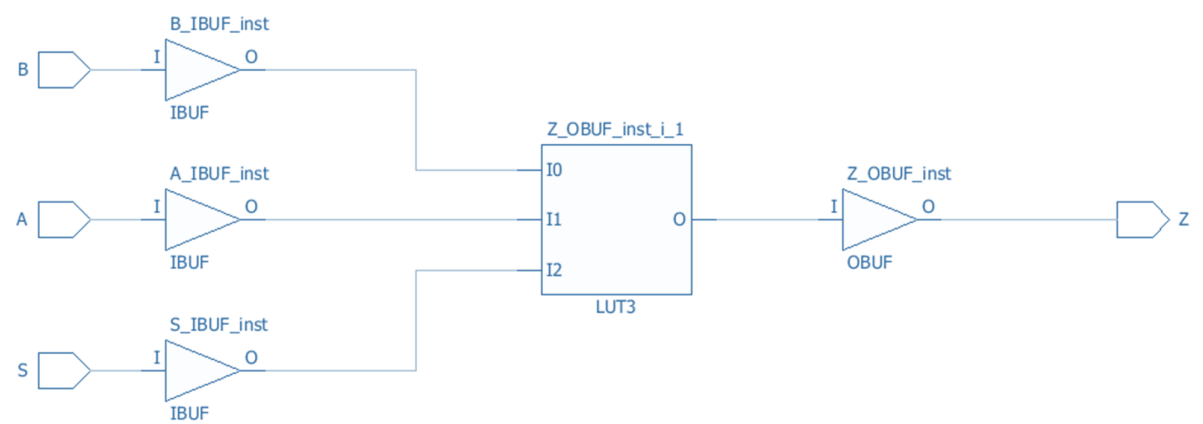
Verilog code for 1-bit two-to-one mux, using the dataflow method

```
module my_2_to_1_mux (input A, B, S, output Z);  
    assign Z = S ? B : A; // assign B to Z if S = 1 or assign A to Z if S = 0;  
endmodule
```

RTL schematic for the 1-bit two-to-one mux



Synthesised design schematic for the 1-bit two-to-one mux



Fill in the truth table for the **LUT3**, as extracted from the synthesised design schematic for the 1-bit two-to-one mux:

Explain how the truth table for the **LUT3** matches that of the truth table indicated in **Figure 2.5**:

GRADED POST-LAB ASSIGNMENT

Complete as much as possible, in **one working bitstream for this whole assignment**. It is much better to have a working program with some completed functionalities, instead of submitting a program without a working bitstream (No marks given).











IMPORTANT CHARACTERS

In this assignment, these are the important characters to note from your student matriculation number:

- The 2nd rightmost numerical value of your student matriculation number (Initialisation, subtask B and subtask C)
- The 3rd rightmost numerical value of your student matriculation number (Subtask A)

INITIALISATION

When the program starts, the seven segment displays must show the following patterns **exactly**, based on the **2nd rightmost numerical value of your student matriculation number**:

2 nd Rightmost Numerical Value	0	1	2	3	4
Required 7-Segments Displays					
2 nd Rightmost Numerical Value	5	6	7	8	9
Required 7-Segments Displays					

SUBTASK A

Based on the **3rd rightmost numerical value of your student matriculation number**, create two separate parallel adders as indicated in the table below:

3 rd Rightmost Numerical Value	Required parallel adders. Each one of these parallel adders must be made up of multiple 1-bit adders	
0	2-bit parallel adder	3-bit parallel adder
1	3-bit parallel adder	2-bit parallel adder
2	2-bit parallel adder	4-bit parallel adder
3	4-bit parallel adder	2-bit parallel adder
4	3-bit parallel adder	4-bit parallel adder
5	4-bit parallel adder	3-bit parallel adder
6	3-bit parallel adder	5-bit parallel adder
7	5-bit parallel adder	3-bit parallel adder
8	2-bit parallel adder	5-bit parallel adder
9	5-bit parallel adder	2-bit parallel adder

Following that, make use of both parallel adders to create a single complete n-bit parallel adder with inputs A and B, and output S. The bits given to the single complete n-bit adder must be divided between the two parallel adders in the following manner:

3 rd Rightmost Numerical Value	<u>Most significant bits</u> of the complete adder must use the following adder	<u>Least significant bits</u> of the complete adder must use the following adder	n-bit input A, or Input B, of the complete adder	n-bit output S of the complete adder
0	2-bit parallel adder	3-bit parallel adder	5 bits	5 bits
1	3-bit parallel adder	2-bit parallel adder	5 bits	5 bits
2	2-bit parallel adder	4-bit parallel adder	6 bits	6 bits
3	4-bit parallel adder	2-bit parallel adder	6 bits	6 bits
4	3-bit parallel adder	4-bit parallel adder	7 bits	7 bits
5	4-bit parallel adder	3-bit parallel adder	7 bits	7 bits
6	3-bit parallel adder	5-bit parallel adder	8 bits	8 bits
7	5-bit parallel adder	3-bit parallel adder	8 bits	8 bits
8	2-bit parallel adder	5-bit parallel adder	7 bits	7 bits
9	5-bit parallel adder	2-bit parallel adder	7 bits	7 bits

The single complete n-bit adder must not have any inputs or outputs for carry bits.

SUBTASK B

When a certain pushbutton, is pressed and held, certain LEDs on the Basys 3 development board must light up. The pushbutton and LEDs to light up are dependent on the **2nd rightmost numerical value of your student matriculation number**, as indicated in the table below:

2 nd Rightmost Numerical Value	Pushbutton to use	LEDs to turn on
0	BTNU (Up)	LD15, LD14, LD13
1	BTNR (Right)	LD15, LD14
2	BTND (Down)	LD15, LD14, LD13, LD12, LD11, LD10
3	BTNL (Left)	LD15, LD14
4	BTNC (Centre)	LD15, LD14, LD13, LD12
5	BTNU (Up)	LD15, LD14
6	BTNR (Right)	LD15, LD14, LD13, LD12
7	BTND (Down)	LD15, LD14, LD13, LD12, LD11, LD10
8	BTNL (Left)	LD15, LD14, LD13, LD12
9	BTNC (Centre)	LD15, LD14, LD13

SUBTASK C

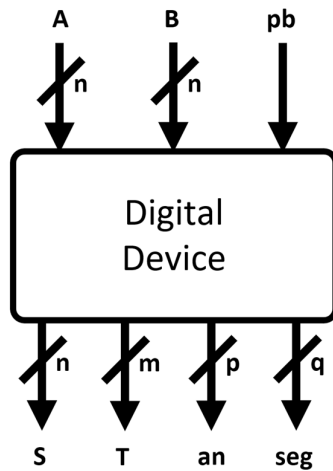
The result from the single complete n-bit adder of SUBTASK A is shown by default on the LEDs of the Basys 3 development board. However, when the user presses and hold the pushbutton that was indicated in SUBTASK B, it is required to have another alternate n-bit result (**AR**) shown on the LEDs of the Basys 3 development board, instead of the n-bit default result (**DR**). That alternate n-bit result is dependent on the **2nd rightmost numerical value of your student matriculation number**, as shown in the table below:

2 nd Rightmost Numerical Value	Alternate result (AR)	Further details
0	3 least significant bits of DR inverted	
1	DR divided by 2	Ignore the decimal point of S (Round down)
2	DR multiplied by 8	Ignore bits that exceed the size of S
3	2 most significant bits of DR inverted	
4	DR divided by 4	Ignore the decimal point of S (Round down)
5	DR multiplied by 2	Ignore bits that exceed the size of S
6	4 least significant bits of DR inverted	
7	DR divided by 8	Ignore the decimal point of S (Round down)
8	DR multiplied by 4	Ignore bits that exceed the size of S
9	3 most significant bits of DR inverted	

Arithmetic operators for addition, subtraction, multiplication, and division are **NOT ALLOWED** anywhere inside this whole graded assignment. No marks will be given if arithmetic operators (+, -, *, /) are used

SUBTASK D (Implementation)

Combine the INITIALISATION, SUBTASK A, SUBTASK B and SUBTASK C together, such that the digital device implemented on the Basys 3 development board only have these ports:



INPUTS:

A : n number of switches (Subtasks A and C)

B : n number of switches (Subtasks A and C)

pb : one pushbutton (Subtask B)

OUTPUTS:

S : n number of LEDs (Subtasks A and C)

T : m number of LEDs (Subtask B)

an : p number of anodes (Initialisation)

seg : q number of segments (Initialisation)

It is compulsory to use the following switches to represent inputs A and B of the Digital Device:

- The input A must use consecutive switches on the Basys 3 development board, with the **least significant bit A[0] linked to SW0**. Similarly, A[1] should be linked to SW1, A[2] to SW2, and so on.
- The input B must use consecutive switches on the Basys 3 Development board, with the **least significant bit B[0] linked to SW8**. Similarly, B[1] should be linked to SW9, B[2] to SW10, and so on.
- Switches not used by A or B must NOT be used or linked to any other signals under any circumstances.** For example, in a 4-bit Digital Device, SW4 to SW7, and SW12 and SW15, should not be linked to any signals, nor used in any circumstances.

It is compulsory to use the following LEDs to represent the output S of the Digital Device:

- The output S must use consecutive LEDs on the Basys 3 development board, with the **least significant bit S[0] linked to LD0**. Similarly, S[1] should be linked to LD1, S[2] should be linked to LD2, and so on.
- LEDs not used by S or T must NOT be used or linked to any other signals under any circumstances.** For example, in a 4-bit Digital Device, with the 2nd rightmost numerical value of the student matriculation number being 3, LD4 to LD13, should not be linked to any signals, nor used in any circumstances.

SUBTASK E (Simulation)

You are also required to simulate and verify any **six unique** test cases in **one single simulation** for the digital device created in SUBTASK D, and which meets the following conditions:

- At least two of the test cases must have the pushbutton (pb) to be ON
- At least two of the test cases must have the pushbutton (pb) to be OFF
- The anode (an) value and segment (seg) value must match the requirement set out in the INITIALISATION

A screenshot (PrintScreen) of the simulation waveform pattern from the Vivado simulation window must be obtained, and then pasted on a 1-page landscape page. Marks are only given if the simulation waveforms are **clear and meet these number representations**:

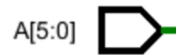
- Input A, input B, input pb, output S, output T are all in **binary representation**
- Output an, output seg, are all in **hexadecimal representation**

SUBTASK F (RTL Schematics)

A screenshot (PrintScreen) of the RTL analysis schematic from Vivado must be obtained, and then pasted on a 1-page landscape page. You are not allowed to do any editing to the RTL analysis schematic screenshot.

For marks to be awarded, the following conditions must be met for the RTL analysis schematics (The images being shown here is for a Digital Device with A, B and S being 6 bits. The 6 bits consist of a 2-bit parallel adder for the 2 least significant bits, and a 4-bit parallel adder for the 4 most significant bits in this example):

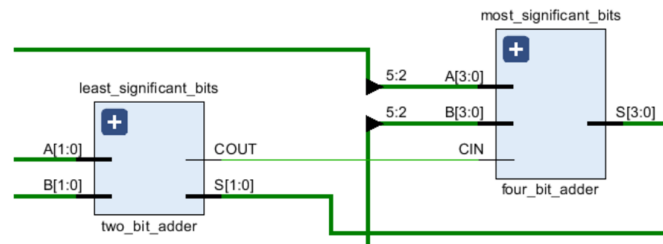
- (1) None of the modules are expanded (That is, screenshot the top-level module clearly after clicking on RTL Analysis -> Schematics). Furthermore, **a minimum of two separate parallel adder modules** (As indicated in SUBTASK A) **must be seen without expanding any modules in the schematics.**
- (2) All inputs of the Digital Device (SUBTASK D) must be clear, including the numerical values between the square brackets. For example:



- (3) All outputs of the Digital Device (SUBTASK D) must be clear, including the numerical values between the square brackets. For example:



- (4) All inputs to, and outputs from, the parallel adder modules must be clear, including the numerical values between the square brackets. For example, in a 6-bit Digital System, there are two parallel adders (As indicated in SUBTASK A) being used:



DOCUMENT UPLOAD REQUIREMENTS:

- (1) Your name and matriculation number on the top of the first page of the document.
- (2) The screenshot (PrintScreen) of the simulation waveform pattern from the Vivado simulation window on the first page of the document.
- (3) The screenshot (PrintScreen) of the RTL analysis schematic from Vivado on the second page of the document.

Print the two pages landscape document as a **single PDF** for LumiNUS upload. The PDF file **must follow the naming template** indicated in the LumiNUS submission instruction at the end of this lab manual.

SUGGESTIONS

- This assignment can be completed by using what you have learnt throughout lab session 2. It is not recommended to use advanced contents not taught in this lab session
- Complete the MUX task and understand the purpose of MUX before working on the assignment
- Use multi-bits to represent the anodes and segment displays
- Assume the pushbutton works like a switch when it is pressed and held
- The following will be taught in subsequent lectures / tutorials / labs, and are thus **not necessary** in lab 2:
 - if-else functions
 - always blocks
 - shift operators

EXAMPLE:

If your student matriculation number is A0159089Y, then:

2nd rightmost numerical value: 8
3rd rightmost numerical value: 0

INITIALISATION



SUBTASK A:

0	2-bit parallel adder	3-bit parallel adder	5 bits	5 bits
---	----------------------	----------------------	--------	--------

SUBTASK B:

8	BTNL (Left)	LD15, LD14, LD13, LD12
---	-------------	------------------------

SUBTASK C:

8	DR multiplied by 4	Ignore bits that exceed the size of S
---	--------------------	---------------------------------------

LUMINUS SUBMISSION INSTRUCTIONS

- Ensure that your bitstream has been successfully generated and tested on your Basys 3 development board **BEFORE** archiving your Vivado workspace for LumiNUS upload
- It is compulsory to archive your project in a compressed form without any saved simulation waveforms. In the uploaded archive, the codes (.v files) are important, not the waveforms (.wdb files). **The archive size should not exceed 4 MB in size for any lab assignments.** Follow the instructions given in the pdf: "Archive Project in Vivado 2018.02"
- **After** following the instructions in "Archive Project in Vivado 2018.02", rename your project archive as indicated in the appendix of this lab manual
- **Separately** from the project archive, **the single PDF document** (Simulation waveform + RTL analysis schematic of the subtractor) is to be named as indicated in the appendix of this lab manual.
- Upload to LumiNUS EE2026 -> Files -> Lab and Project - Materials and Submissions -> Lab 2 Submission
- Download your LumiNUS archive after uploading. **Click and drag the single folder within that archive to desktop, and then open the Vivado project in that extracted folder to see if it can be opened. Check if you can also run your bitstream correctly.** No project files and no working bitstream is equivalent to losing all marks
- The LumiNUS upload must be completed by **Wednesday 9th February 2022, 12:00 P.M. (Noon)**. Avoid uploading during the grace period of 2 hours
- A penalty of 25% applies for late submissions of up to 1 week.
- The late submission folder closes 1 week after the original deadline. **Late submissions are not accepted and not graded if a submission is found within the on-time folder, or if grading has already started on an earlier submitted file.** The late submission folder will be located at: LumiNUS EE2026 -> Files -> Lab and Project - Materials and Submissions -> Lab 2 Submission (Late Submission)

Plagiarism is penalised with a 100% penalty for all SOURCES and RECIPIENTS

All past and future submissions, and marks, will be reviewed in greater detail, for any person found to have plagiarised

ALL THE SUBMISSION INSTRUCTIONS LISTED ABOVE WILL AFFECT YOUR GRADES!

GRADING PROCESS

- During subsequent lab sessions, our graders will be providing you updates on the grading of your submission
- Submissions not following all the **LUMINUS SUBMISSION INSTRUCTIONS** (listed above) will not be graded immediately, and will instead be reviewed towards the end of the semester. **You will not be able to see your results during the labs in such situations**

APPENDIX (COMPULSORY renaming before just LumiNUS upload):

It is **compulsory to rename your project archive and PDF report**, just before the LumiNUS upload, as listed in the table below.

Do not change any other part of the naming. Simply **copy** the naming from the table below, and **paste** it while renaming your project archive.

Penalties will be incurred if your submission cannot be found according to the exact naming template below.

Archive Naming	Report Naming
L2_Thurs_AM_Aaron Chan Zhi_476_Archive	L2_Thurs_AM_Aaron Chan Zhi_476_Report
L2_Thurs_AM_Ajay Shanker_806_Archive	L2_Thurs_AM_Ajay Shanker_806_Report
L2_Thurs_AM_Alphonsus Teow Rui Jie_502_Archive	L2_Thurs_AM_Alphonsus Teow Rui Jie_502_Report
L2_Thurs_AM_Alvin Ben Abraham_394_Archive	L2_Thurs_AM_Alvin Ben Abraham_394_Report
L2_Thurs_AM_Amadeus Lim Ding Shin_412_Archive	L2_Thurs_AM_Amadeus Lim Ding Shin_412_Report
L2_Thurs_AM_Amit Rahman_599_Archive	L2_Thurs_AM_Amit Rahman_599_Report
L2_Thurs_AM_Ang Jia Le Marcus_025_Archive	L2_Thurs_AM_Ang Jia Le Marcus_025_Report
L2_Thurs_AM_Chan Ee Hong_898_Archive	L2_Thurs_AM_Chan Ee Hong_898_Report
L2_Thurs_AM_Chen Zi Han_549_Archive	L2_Thurs_AM_Chen Zi Han_549_Report
L2_Thurs_AM_Chien Jing Wei_540_Archive	L2_Thurs_AM_Chien Jing Wei_540_Report
L2_Thurs_AM_CHUA WEI XUAN_716_Archive	L2_Thurs_AM_CHUA WEI XUAN_716_Report
L2_Thurs_AM_Chua Wen Xin Kyrene_431_Archive	L2_Thurs_AM_Chua Wen Xin Kyrene_431_Report
L2_Thurs_AM_Darren Loh Rui Jie_289_Archive	L2_Thurs_AM_Darren Loh Rui Jie_289_Report
L2_Thurs_AM_Dennis Wong Guan Ming_806_Archive	L2_Thurs_AM_Dennis Wong Guan Ming_806_Report
L2_Thurs_AM_Huang Yu Chiao_102_Archive	L2_Thurs_AM_Huang Yu Chiao_102_Report
L2_Thurs_AM_Ivan Theng Wen Rong_344_Archive	L2_Thurs_AM_Ivan Theng Wen Rong_344_Report
L2_Thurs_AM_Jia Yixuan_150_Archive	L2_Thurs_AM_Jia Yixuan_150_Report
L2_Thurs_AM_Karthikeyan Vigneshram_697_Archive	L2_Thurs_AM_Karthikeyan Vigneshram_697_Report
L2_Thurs_AM_Leong Wei Lun Alfred_609_Archive	L2_Thurs_AM_Leong Wei Lun Alfred_609_Report
L2_Thurs_AM_Liu Junhao_523_Archive	L2_Thurs_AM_Liu Junhao_523_Report
L2_Thurs_AM_Marvin Pranajaya_683_Archive	L2_Thurs_AM_Marvin Pranajaya_683_Report
L2_Thurs_AM_Ng Sihan Ian_817_Archive	L2_Thurs_AM_Ng Sihan Ian_817_Report
L2_Thurs_AM_Ong Zhi Hong_922_Archive	L2_Thurs_AM_Ong Zhi Hong_922_Report
L2_Thurs_AM_Raymond Bala_127_Archive	L2_Thurs_AM_Raymond Bala_127_Report
L2_Thurs_AM_See Zhuo Rui Jorelle_490_Archive	L2_Thurs_AM_See Zhuo Rui Jorelle_490_Report
L2_Thurs_AM_Shanmugam Surya_189_Archive	L2_Thurs_AM_Shanmugam Surya_189_Report
L2_Thurs_AM_Shawn Tan Jinhui_247_Archive	L2_Thurs_AM_Shawn Tan Jinhui_247_Report
L2_Thurs_AM_Shin Donghun_808_Archive	L2_Thurs_AM_Shin Donghun_808_Report
L2_Thurs_AM_Stefan Choo Bin Hao_098_Archive	L2_Thurs_AM_Stefan Choo Bin Hao_098_Report
L2_Thurs_AM_Syed Muhamad Amali B Syed_373_Archive	L2_Thurs_AM_Syed Muhamad Amali B Syed_373_Report
L2_Thurs_AM_Teh ZiChun_328_Archive	L2_Thurs_AM_Teh ZiChun_328_Report
L2_Thurs_AM_Wen Chen Yu_109_Archive	L2_Thurs_AM_Wen Chen Yu_109_Report
L2_Thurs_AM_Wilson Ng Jing An_686_Archive	L2_Thurs_AM_Wilson Ng Jing An_686_Report
L2_Thurs_AM_Yong Chin Han_814_Archive	L2_Thurs_AM_Yong Chin Han_814_Report
L2_Thurs_AM_Yuan Xinrui_211_Archive	L2_Thurs_AM_Yuan Xinrui_211_Report