EE2026: DIGITAL DESIGN

Academic Year 2020-2021, Semester 1

LAB 1: Quick Start Guide to Vivado 2018.2, Basys 3 Development Board, and Verilog HDL

FOR ALL EE2026 LAB AND PROJECT SESSIONS [VENUE: Digital Electronics Lab E4-03-07]:

- You are **highly encouraged to bring to lab, your own laptop with Vivado 2018.2 already installed**. You may still use the desktop PC in lab if you do not have a laptop that can be brought to lab.

- Use the **D:\MyWork** folder for your work if you are using the lab PC. You are required to **delete** all folders within the **D:\MyWork** folder before starting your lab session.

- **Delete** your work folder from the laboratory's computers after your session is over. You are responsible to **safeguard** your confidential programs. For assessable programs, you will be penalised if two programs with similarities beyond empirical evidence are detected. Both the source(s) and recipient(s) of plagiarised programs are equally penalised.

- All lab sessions require that you have **carefully reviewed the relevant lecture and tutorial materials before attendance**. Contents taught during the theory classes, with emphasis on the Verilog language and structure, will directly be applied to solve practical problems during the lab sessions.

OVERVIEW:

Using a simple Boolean design problem, an introductory approach to the Vivado software used in EE2026 will be covered. Quick instructions on downloading and installing the Vivado software on your personal computer are provided. The Vivado software is a comprehensive integrated development environment (IDE) for FPGA design flow.

In this lab:

- An introduction to very basic Verilog HDL (Hardware Description Language) is provided.
- The overall process flow of designing, synthesising, simulating and implementing a program is covered.
- Programming Digilent's Basys 3 development board, which features an FPGA from Xilinx's Artix-7 family, is illustrated.

GRADED ASSIGNMENT [LUMINUS SUBMISSION: TUESDAY 1st SEPTEMBER 2020, NOON]:

- To display a specific character on the 7-segment display(s) of the Basys 3 development board if a password is correct

Further details are available at the end of this lab manual.

VIVADO DOWNLOAD AND INSTALLATION:

The Vivado 2018.2 software is already installed on the computers in the Digital Electronics Lab, and are ready for immediate usage. **It is also required that you install such software on your own personal computer, preferably before coming for the first lab session.** Some quick guidelines on installing the required software for EE2026 on your personal computer is provided in this section.

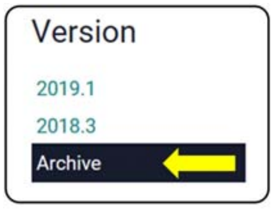
Software Weblink

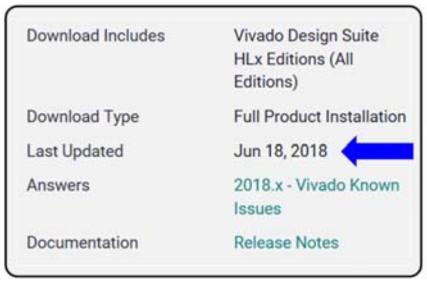
https://www.xilinx.com/support/download.html

Software Name [Mac version is not supported]

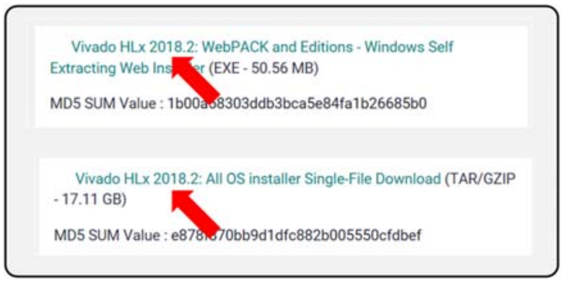
Vivado Design Suite - HLx Editions - 2018.2 [Last Updated: Jun 18, 2018]

Warning: *Do not use other versions of the software. Only the* ***Vivado 2018.2*** *Windows version has been tested. Computer compatibility issues will occur with other versions of the software, and assessment of your project may not be possible. This will lead to loss of project marks if your project cannot be assessed.*





Download Includes	Vivado Design Suite HLx Editions (All Editions)
Download Type	Full Product Installation
Last Updated	Jun 18, 2018
Answers	2018.x - Vivado Known Issues
Documentation	Release Notes



Vivado HLx 2018.2: WebPACK and Editions - Windows Self Extracting Web Installer (EXE - 50.56 MB)

MD5 SUM Value : 1b00a68303ddb3bca5e84fa1b26685b0

Vivado HLx 2018.2: All OS installer Single-File Download (TAR/GZIP - 17.11 GB)

MD5 SUM Value : e878a70bb9d1dfc882b005550cfdbef

Select either one of the two available installers for download, based on your preference:

- Vivado HLx **2018.2**: WebPACK and Editions - Windows Self Extracting Web Installer (EXE - 50.56 MB)
- Vivado HLx **2018.2**: All OS installer Single-File Download (TAR/GZIP - 17.11 GB)

Registration is required for any downloads from the Xilinx website, but not required for installation and program usage.

Installation

During the installation phase, you will be given an option on the edition to install. The edition to be installed is:

- Vivado HL WebPACK

For subsequent customisation options, you can leave it to the default settings.

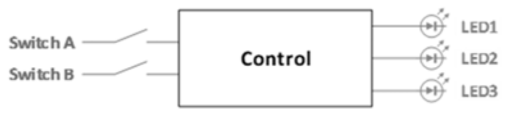
Post-Installation

Restart your computer before using the Vivado 2018.2 software. You may wish to uninstall the Xilinx Information Centre from the Windows control panel as it is not needed. This will prevent unnecessary pop-up messages by Xilinx from appearing.

DESCRIPTION OF THE SIMPLE BOOLEAN DESIGN TASK

The following task is required to be implemented on the Basys 3 development board:

- When switch **A** turns on, only **LED1** lights up.
- When switch **B** turns on, only **LED2** lights up.
- When both switches **A** and **B** turn on, **LED1**, **LED2**, and **LED3** light up.



UNDERSTANDING | TASK 1

Complete the truth table for the simple boolean design task:

INPUT		OUTPUT			MINTERM
A	B	LED1	LED2	LED3	
0	0				$\bar{A}\bar{B}$
0	1				$\bar{A}B$
1	0				$A\bar{B}$
1	1				AB

Deriving an SOP Boolean Equation for the Design Task

Given any truth table with any number of input variables, the sum-of-products (SOP) or product-of-sums (POS) form may be used to write out a Boolean equation for each output variable. Let us use the canonical SOP form for **LED1**:

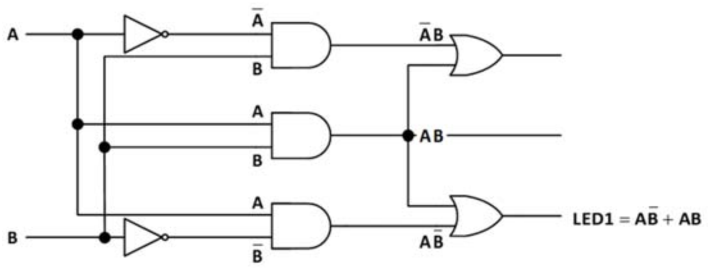
$$\mathbf{LED1} = \boldsymbol{A\bar{B} + AB}$$

UNDERSTANDING | TASK 2

Work out the canonical SOP Boolean equations for **LED2** and **LED3**

Illustrating Logic Expressions by Using a Schematic of Gates

The Boolean equations for **LED1**, **LED2**, and **LED3**, can be implement by using: 2 **NOT** gates, 3 **AND** gates, 2 **OR** gates



Verilog Hardware Description and FPGA Implementation

Xilinx's Vivado software is an integrated design environment that has numerous amounts of advanced features used in the industry, and among which we will be introducing the following:

- Writing and editing HDL codes for digital system designs.
- Simulation of the design's behaviour.
- Synthesis of the codes, in order to convert the design from textual description into logic gates.
- Implementation of the design to map and route the logic to a target FPGA.
- Optimising the synthesis, implementation, and bitstream generation according to the user's strategies. The default optimisation strategies shall be used in EE2026, as changing them is beyond the scope of introductory digital designs.
- Programming an FPGA with the optimised bitstream.

The remaining part of this lab manual will now briefly show the general steps required to go from the design task, to the FPGA implementation on the Basys 3 development board, for EE2026 purposes.

INTRODUCTORY QUICK START GUIDE TO XILINX'S VIVADO 2018.2 SOFTWARE

During your lab session, your EE2026 graduate and lab assistants may provide you helpful hints on the usage of the Vivado 2018.2 software, beyond the most basic things that are described in this section.

Creating a New Verilog Project in Vivado

Start Menu: Open the executable: Vivado 2018.2. You will need to wait multiple seconds before the program opens

Quick Start: Select **Create Project** and continue

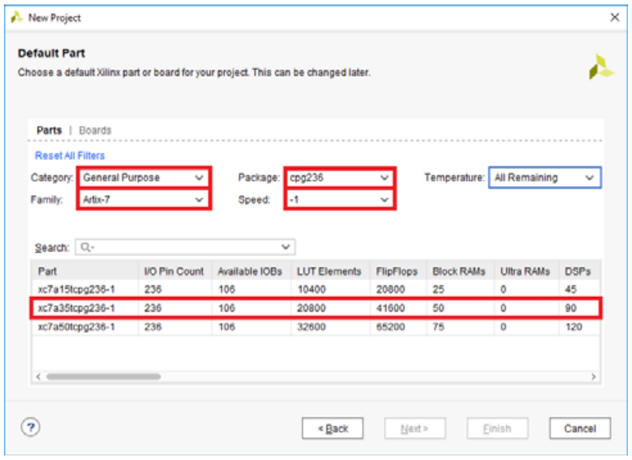
Project Name: Enter a **Project name** and **Project Location**. Ensure that the **Project name** and complete **Project location** for your project folder <u>does not have any spaces or special characters</u>, and that your **Project name** <u>does not start with a number</u>

Project Type: Select **RTL Project**, and uncheck "**Do not specify sources at this time**"

Add sources:
- **Create File.** File type is Verilog. Example: simple_boolean
- **Target language:** Verilog. **Simulator language:** Mixed

Add constraints (optional): Click on next without any changes

Default Part: Specify the FPGA chip that will be used. The Basys 3 development board uses the **xc7a35tcpg236-1** chip



Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

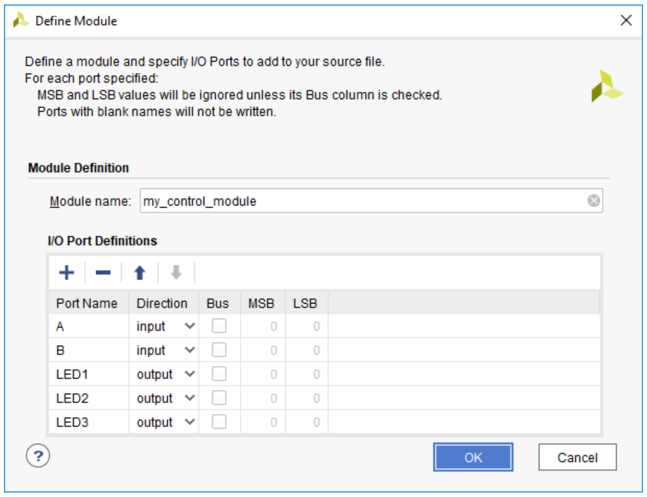
Category: General Purpose
Family: Artix-7
Package: cpg236
Speed: -1
Temperature: All Remaining

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs
xc7a15tcpg236-1	236	106	10400	20800	25	0	45
xc7a35tcpg236-1	236	106	20800	41600	50	0	90
xc7a50tcpg236-1	236	106	32600	65200	75	0	120

New Project Summary: To create the project, click **Finish**

Define Module: A module, that is contained within the file, need top be created. Create one based on the inputs and outputs of the simple boolean design task.



Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Module name: my_control_module

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
A	input	☐	0	0
B	input	☐	0	0
LED1	output	☐	0	0
LED2	output	☐	0	0
LED3	output	☐	0	0

Using Vivado Text Editor to Write Verilog HDL Code

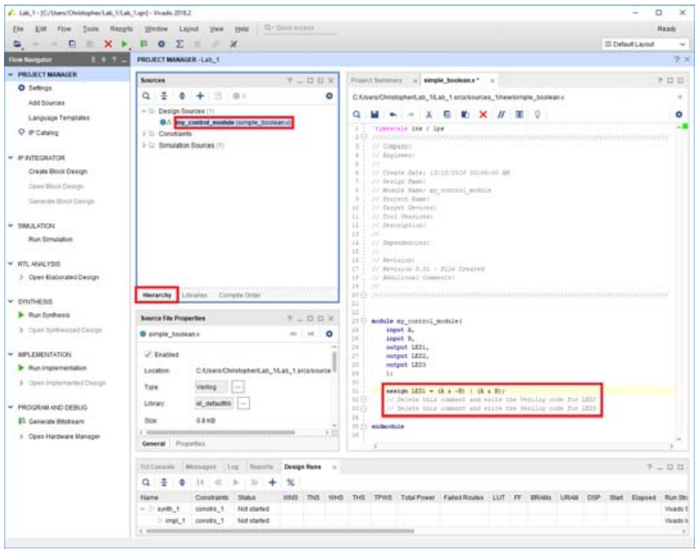
Open the module that has been created by double clicking on it in the Sources window

UNDERSTANDING | TASK 3

Code the behaviour of the module by converting the SOP expressions for **LED1**, **LED2**, and **LED3** to the Verilog equivalent. The codes are to be inserted between the keywords **module** and **endmodule**.

Some Verilog representation of common operators are as tabulated below:

Operators		Verilog Representation
OR	$\mathbf{A + B}$	|
AND	$\mathbf{AB}$	&
NOT	$\mathbf{\bar{A}}$	~
XOR	$\mathbf{A \oplus B}$	^



```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: [illegible]
7  // Design Name:
8  // Module Name: my_control_module
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21
22
23 module my_control_module(
24     input A,
25     input B,
26     output LED1,
27     output LED2,
28     output LED3
29     );
30
31     assign LED1 = (A & ~B) | (A & B);
32     // Delete this comment and write the Verilog code for LED2
33     // Delete this comment and write the Verilog code for LED3
34
35 endmodule
36
```

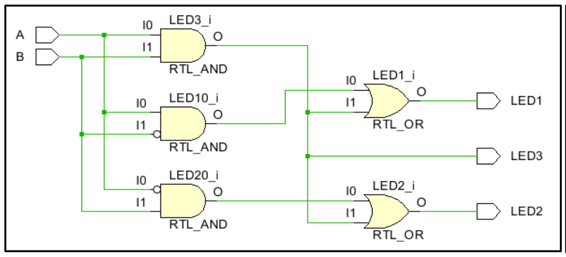
The **assign** statement causes the left hand side of the expression to be updated *every* time there is a change on the right hand side of the expression. It is therefore called a *continuous assignment* statement, describing combinational logic whereby the output on the left is a function of current inputs on the right.

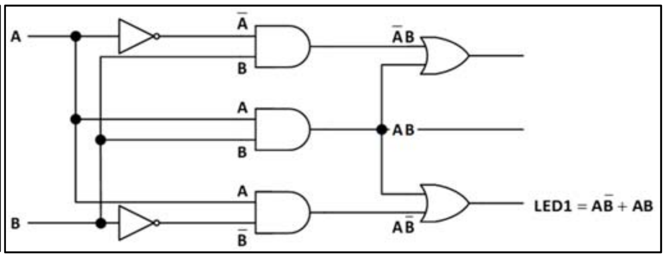
The statements on line 31 till line 33 execute concurrently. This is in contrast to sequential execution of statements in a computer programming language such as C, or procedural assignment that will be taught in subsequent lab sessions.

Save your current file by clicking on **File → Save File**, or by pressing **Ctrl+S**. Each time a file is saved, a syntax check is carried out. After saving, perform the following: In the **Flow Navigator** window, under **RTL ANALYSIS**: **Open Elaborated Design**, select **Schematic.** The schematics window will appear, showing the Register Transfer Level (RTL) schematic of the design.

UNDERSTANDING | TASK 4

What similarities and differences do you notice between the RTL schematic and the schematic obtained from the previous section. How do they compare to the actual schematic obtained on your computer screen?



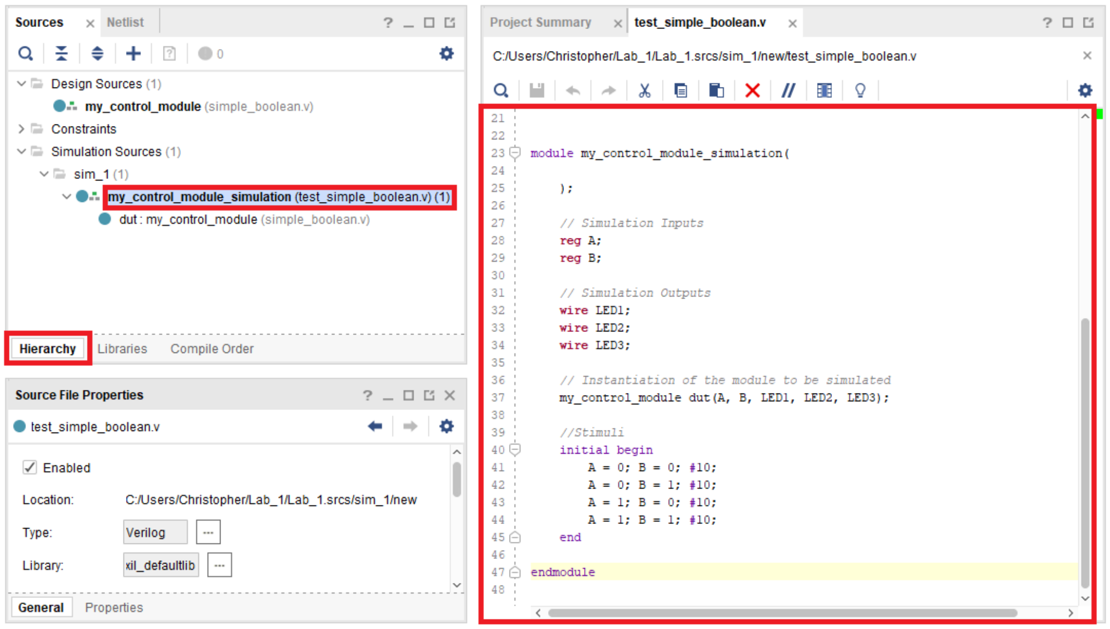


Testbench and Behavioural Simulation

After writing the codes, there is a need to test them to check their behaviours. Inputs are applied to a module, and the outputs are checked to verify whether the module operates as intended. A testbench is an HDL module that is used to test another module. In this example, a testbench will be created to apply inputs to the module to be tested:

- From the **PROJECT MANAGER**, click on **Add Sources**, followed by **Add or create simulation sources**
- **Create File**, and provide a Verilog file name, such as **test_simple_boolean**
- In the subsequent **Define Module** window, provide a **Module name**, such as **my_control_module_simulation**
- Do not input any **I/O Port Definitions**, and click on **OK** to finish creating the simulation module template

From the Sources window, open the simulation file. Then, within the simulation module, provide the following codes and save them, with the final screenshot looking similar to the image shown below:



Sources | Netlist

```
Design Sources (1)
    my_control_module (simple_boolean.v)
Constraints
Simulation Sources (1)
    sim_1 (1)
        my_control_module_simulation (test_simple_boolean.v) (1)
            dut : my_control_module (simple_boolean.v)
```

Hierarchy | Libraries | Compile Order

Source File Properties

test_simple_boolean.v

☑ Enabled

Location:	C:/Users/Christopher/Lab_1/Lab_1.srcs/sim_1/new
Type:	Verilog
Library:	xil_defaultlib

General | Properties

Project Summary | test_simple_boolean.v

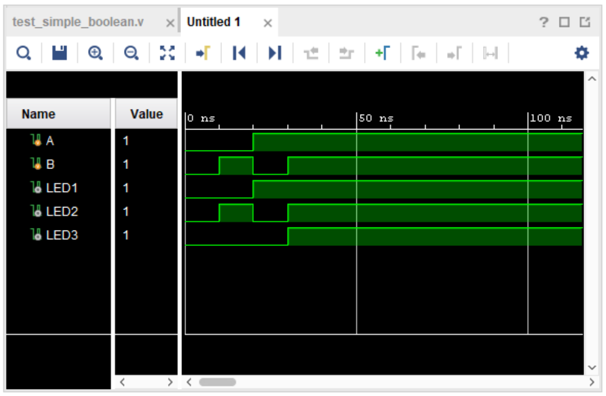
C:/Users/Christopher/Lab_1/Lab_1.srcs/sim_1/new/test_simple_boolean.v

```
module my_control_module_simulation(

    );

    // Simulation Inputs
    reg A;
    reg B;

    // Simulation Outputs
    wire LED1;
    wire LED2;
    wire LED3;

    // Instantiation of the module to be simulated
    my_control_module dut(A, B, LED1, LED2, LED3);

    //Stimuli
    initial begin
        A = 0; B = 0; #10;
        A = 0; B = 1; #10;
        A = 1; B = 0; #10;
        A = 1; B = 1; #10;
    end

endmodule
```

If there are no syntax errors, in the **Flow Navigator** window, under **SIMULATION**, select **Run Simulation**, followed by **Run Behavioural Simulation** in order to create the simulation waveform window.

A noticeable waveform pattern may not be seen by default, as the time resolution used in the simulation is very small as compared to the amount of time the simulation is ran. Hence, with the simulation windows being the active window and from the menu, select **View → Zoom Fit**, or press **Ctrl+0**

Look at the simulation results closely. How do the waveforms show that your design is indeed working as desired? Consider trying out the various options provided in the simulation window before going back to the Workspace. Do not save the simulation window waveform, as this consumes a large amount of storage space.



Name	Value
A	1
B	1
LED1	1
LED2	1
LED3	1

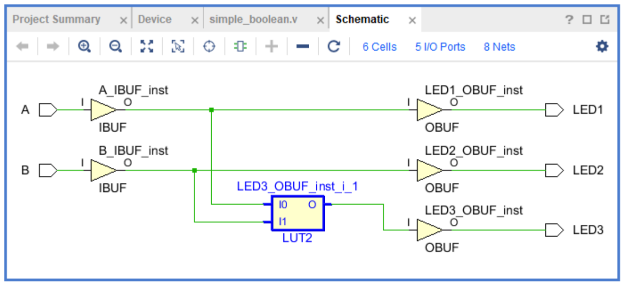
Synthesis

Logic synthesis transforms HDL code into an optimised set of logic gates to reduce the amount of hardware, and to efficiently perform the intended function.

Right-click on your Verilog design source file and select **Set as Top**. This option is disabled if the file is already the top module, and in such a case, proceed directly to the next step. In general, when there are multiple design and simulation modules, the "Set as Top" option selects the design, or simulation, modules to be considered when performing the different stages of the project flow.

In the **Flow Navigator** window, under **SYNTHESIS**, select **Run Synthesis**. While Vivado performs synthesis, the Project Status Bar at the top right provides an indication of the ongoing progress.

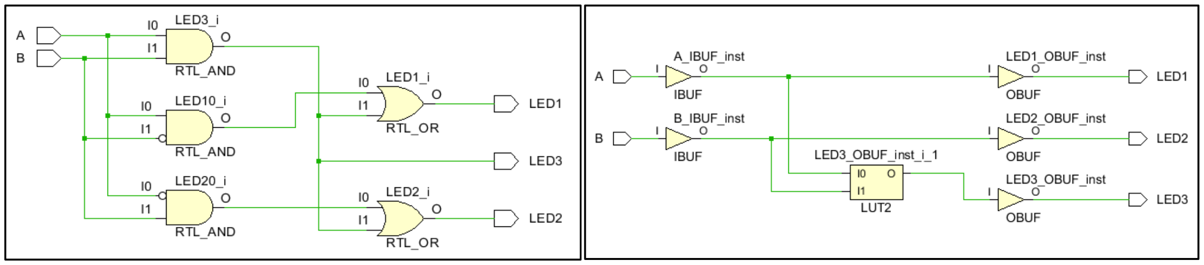
After the synthesis has been successfully completed, in the **Flow Navigator** window, under **SYNTHESIS**, expand **Open Synthesized Design**, and select **Schematics**. The schematic of the synthesised design will be generated and this synthesised circuit is an optimised version of the RTL schematic that was obtained



Click on the Look-up Table (LUT) that defines how the output LED3 behaves. The **Cell Properties** window will appear for that specific LUT. In the **Cell Properties** window for the LUT of **LED3**, open the **Truth Table** tab. Notice how for this simple example, this LUT is behaving as a simple AND gate.

UNDERSTANDING | TASK 5

Compare the optimised and non-optimised schematics. How is this optimised circuit equivalent to the SOP equations of the simple boolean design task?

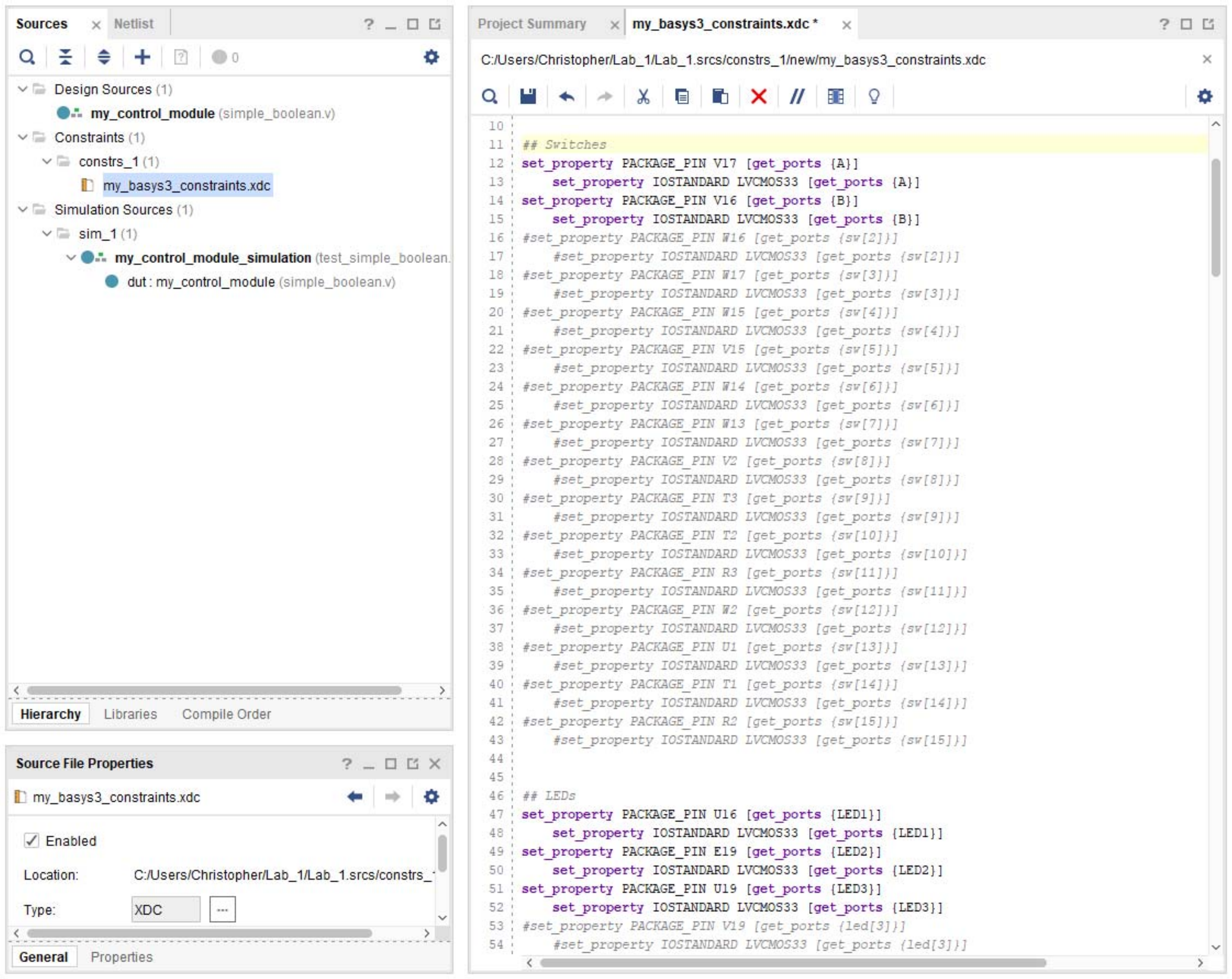


Design Constraints

Design constraints, such as timing and physical I/O pin mapping, must be defined before doing an implementation, following which the program can be downloaded to the FPGA device. Proceed with the following sequence:

- Expand **PROJECT MANAGER** in the **Flow Navigator** panel, and **click Add Sources**
- Select **Add or create constraints** and click **Next**
- Click on **Create File** and give the XDC file a file name, such as **my_basys3_constraints**. The XDC format stands for Xilinx Design Constraints here
- Open the **my_basys3_constraints.xdc** file from the **Sources** window. It will be an empty .xdc file.
- A template, known as the **Basys3_Master.xdc** is provided. Open that template using a basic text editor, such as notepad.
- Copy all the contents from that template to your **my_basys3_constraints.xdc**. All the lines are commented out by default.
- Link the signals (A, B, LED1, LED2, LED3) of your design, to some physical pins of the FPGA, by uncommenting relevant lines. Input signals can be linked to switches, whereas the output signals can be linked to LEDs, on the Basys3 development board.

An example of the above steps is shown below:


Sources
Netlist
Design Sources (1)
my_control_module (simple_boolean.v)
Constraints (1)
constrs_1 (1)
my_basys3_constraints.xdc
Simulation Sources (1)
sim_1 (1)
my_control_module_simulation (test_simple_boolean.
dut : my_control_module (simple_boolean.v)
Hierarchy
Libraries
Compile Order
Source File Properties
my_basys3_constraints.xdc
Enabled
Location:
C:/Users/Christopher/Lab_1/Lab_1.srcs/constrs_
Type:
XDC
General
Properties
Project Summary
my_basys3_constraints.xdc *
C:/Users/Christopher/Lab_1/Lab_1.srcs/constrs_1/new/my_basys3_constraints.xdc

```

## Switches
set_property PACKAGE_PIN V17 [get_ports {A}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A}]
set_property PACKAGE_PIN V16 [get_ports {B}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B}]
#set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
#set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
#set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
#set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]


## LEDs
set_property PACKAGE_PIN U16 [get_ports {LED1}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED1}]
set_property PACKAGE_PIN E19 [get_ports {LED2}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED2}]
set_property PACKAGE_PIN U19 [get_ports {LED3}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED3}]
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
```

Implementation, Bitstream Generation and Program Download

The implementation phase will map the design to available physical resources on the FPGA hardware. In the **Flow Navigator** window, under **IMPLEMENTATION**, select **Run Implementation**. This will make use of the design constraint file that had been created earlier on.

After the implementation phase, there is a need to generate a file that can be downloaded to the FPGA. Such a file is called a bitstream file, and it consists of binary values 0's and 1's that tells the FPGA how to behave. In the **Flow Navigator** window, under **PROGRAM AND DEBUG**, select **Generate Bitstream**. A successful bitstream generation is the last step required before downloading the program to the FPGA.

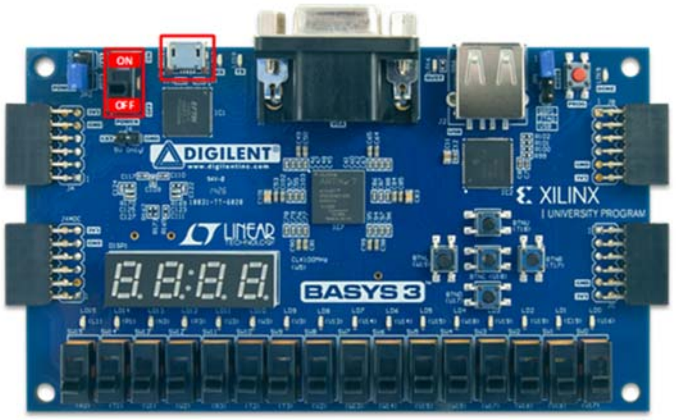
Before using your Basys 3 development board, and to prevent potential damage to it, take note of the following recommendations and warnings to extend the longevity of the device:

⚠ Make sure the Basys 3 development board is powered OFF by placing SW16 in the OFF position before connection to/removal from the USB port of the computer.

⚠ Do not force in the micro-USB cable upside down to the Basys 3 development board, as this will damage your micro-USB port and device. **Carefully connect to the micro-USB cable in the correct orientation.**
[**Common cause of board damage in EE2026 labs – Students are required to buy replacements in cases of negligence**]

⚠ The chips on the board are electrostatic sensitive. Avoid touching them. Handle the board by the edges to prevent damage.

⚠ Make sure the board is not in contact with any metal components, whether above or below. Do not place any liquid sources near the FPGA board.



After connection of the Basys 3 development board to the computer, turn on the power by setting SW16 in the ON position. Test the functionality of your Basys 3 development board before downloading any program to it, according to instructions that will be provided during your lab session. If confirmed to be working, proceed with the following steps:

- Expand **Program and Debug** in the **Flow Navigator Panel**
- Expand **Open Hardware Manager**
- Click **Open Target**
- Select **Auto Connect.** In case connection fails, consider pressing the 'reset' button, or turn your device OFF for a few seconds and ON again, while ensuring that it is detected and installed on your computer. Then try **Auto Connect** again
- If successful, the **Program Device** will be enabled, and you will be able to select **xc7a35t_0**
- By default, if the bitstream was successfully generated, the path name in the **Bitstream file** is automatically provided
- Download the .bit file to the FPGA by clicking on **Program**

UNDERSTANDING | TASK 6

Your program will then be downloaded to the FPGA. Verify the functionality of the design by using the input devices you have assigned to A, B, and observing the output devices assigned to LED1, LED2 and LED3. Check what happens if the 'reset' pushbutton on the Basys 3 development board is pressed, or if power is loss for a short amount of time.

CLOSING NOTES FOR LAB 1

Now that you have successfully completed your FPGA design flow, one final practice task is provided to you for completion before ending the lab session. This practice task is not graded, but you need to inform your guiding G.A. of the task completion.

FINAL UNDERSTANDING | PRACTICE TASK FOR LAB 1

- **Create a new Vivado project from scratch. Do not reuse your existing project or design**
- The same design as described for the simple boolean design task need to be implemented, with the following exception: There is an additional switch C, and if this switch C is in the OFF state, it forces all the three LEDs to be in the OFF state. If the switch C is in the ON state, the design behaves exactly as described for the simple boolean design task. The switch C is to be mapped to SW[*Your birthday month + 3*] on the Basys 3 development board
- Simulate your design, as well as implement it on the Basys 3 development board

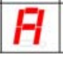
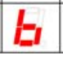
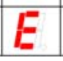
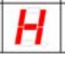
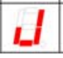
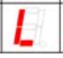
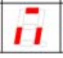
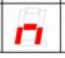
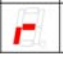
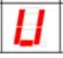
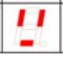
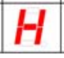
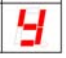
GRADED POST-LAB ASSIGNMENT

ASSIGNMENT

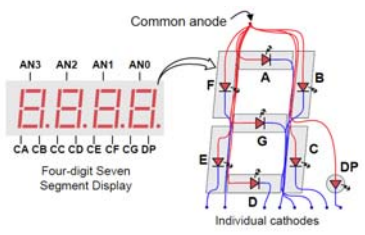
Based on your student matriculation number you are required to display a specific character on specific seven-segment displays. After you have done so, modify your code such that the latter only happens when a 10-bit value (password) is correctly set through the usage of 10 switches: SW0 to SW9. If the 10-bit value is wrong, all the seven-segment displays do not show anything.

REQUIREMENT BASED ON YOUR STUDENT MATRICULATION NUMBER

The character to display is tabulated below:

Last character of your student matriculation number	A	B	E	H	J	L	M	N	R	U	W	X	Y
Required character on the seven-segment display													

There are 7 LED segments in each display, with an additional decimal point. They are respectively denoted by "seg[0]" to "seg[6]", and "dp", in the Basys_Master.xdc constraint file.



There are 4 seven-segment displays on the Basys 3 development board. Each one of the displays is controlled by a common anode pin, thus resulting in a total of 4 common anodes. These active-low pins are denoted as "an[3]" to "an[0]" in the Basys_Master.xdc constraint file. (For more information, you can refer to the Basys 3 reference manual, pages 14 to 16)

Based on the rightmost numerical value of your student matriculation number, hardcode the values according to the table below:

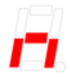
Last numerical value of your student matriculation number	AN3	AN2	AN1	AN0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

The correct password that allows the seven-segment displays to show character(s) is determined as follows:
- From your student matriculation number, consider the 5 rightmost numerical digits (Ignore the alphabet).
- These 5 digits will represent the active-high switches that need to be ON, while all the other active-high switches between SW0 to SW9 must be OFF, in order to be considered a correct password.

If the password is incorrect, it is compulsory for all the seven-segment displays to not show anything.

EXAMPLE 1 OF REQUIRED DISPLAYS BASED ON YOUR STUDENT MATRICULATION NUMBER

For example, if your student matriculation number is A3456789N, then it means that the character to appear on the display is:

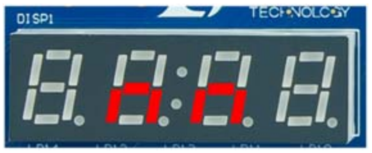


And since the rightmost numerical value of the student matriculation number is 9, AN3 will be 1, AN2 will be 0, AN1 will be 0, and AN0 will be 1. A value of 0 to a common anode makes the specific display active. Hence, with a value of 1001 to the 4 common anodes, the two middle displays will be active (on).

Furthermore, the five rightmost numerical values of the student matriculation number are: 5, 6, 7, 8, 9. Hence, the correct password that allows the seven-segment displays to be on would be:

SW9	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF	OFF

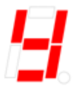
The following value should appear on DISP1 (4 seven-segment displays) when the password is correct:



When the password is wrong (not having the exact 10 switches as tabulated above to be ON/OFF) nothing is displayed on DISP1.

EXAMPLE 2 OF REQUIRED DISPLAYS BASED ON YOUR STUDENT MATRICULATION NUMBER

For example, if your student matriculation number is A0125522Y, then it means that the character to appear on the display is:

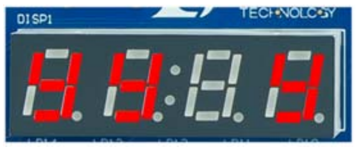


And since the rightmost numerical value of the student matriculation number is 2, AN3 will be 0, AN2 will be 0, AN1 will be 1, and AN0 will be 0. A value of 0 to a common anode makes the specific display active. Hence, with a value of 0010 to the 4 common anodes, the second display from the right will be inactive (off).

Furthermore, the five rightmost numerical values of the student matriculation number are: 2, 5, 5, 2, 2. Hence, the correct password that allows the seven-segment displays to be on would be:

SW9	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
OFF	OFF	OFF	OFF	ON	OFF	OFF	ON	OFF	OFF

The following value should appear on DISP1 (4 seven-segment displays) when the password is correct:



When the password is wrong (not having the exact 10 switches as tabulated above to be ON/OFF) nothing is displayed on DISP1.

HINTS

- Create a new Vivado project for this assignment, instead of continuing from your previous Vivado project
- This assignment can be fully completed by using only what you have learnt throughout lab session 1. It is not recommended to use contents not taught in this lab session, such as multi-bit assignments (Eg. Do not use assign AN[2:1] = 3)
- **Do not use if-else functions. This is not covered in lab 1**
- Consider each one of the seven segments as being one active-low LED, and hardcode it as '0' or '1'
- Consider each one of the four displays as being an active-low "enable command", and hardcode it as '0' or '1'
- All four displays, as well as all the seven segments (and 1 decimal point led) need to be given a value and constrained
- It is a good practice to code and test each functionality separately, before combining them all to achieve all requirements
- **You will need DISP1 and the ability to program for combinational circuits in later labs. Ensure you understand this part well**

LUMINUS SUBMISSION INSTRUCTIONS

- Complete as much required functionalities as possible within the given deadline, and ensure that your bitstream has been successfully generated and tested on your Basys 3 development board **BEFORE** archiving your Vivado workspace for LumiNUS upload. No working bitstream is equivalent to no marks (It is best to have some working functionalities / requirements, instead of not having any bitstream at all while trying all requirements)

- It is compulsory to archive your project in a compressed form without any simulation waveforms. In the uploaded archive, the codes (.v files) are important, not the waveforms (.wdb files). **The archive size should not exceed 1 MB in size for lab 4.** 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.

- Upload to LumiNUS EE2026 -> Files -> Lab and Project - Materials and Submissions -> Lab 1 Submission

- Download your LumiNUS archive after uploading. **Unzip it / Extract all, and check if you can run your bitstream correctly**. No project files and no working bitstream is equivalent to losing all marks

- The LumiNUS upload must be completed by **Tuesday 1st September 2020, 12:00 P.M. (Noon).** Do not plan to upload 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 if you have already submitted on time, 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 1 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 they will instead be reviewed towards the end of the semester. **You will not be able to see your results during the lab sessions in such situations**

APPENDIX (Renaming submission just before LumiNUS upload):

It is **compulsory to rename your project archive**, just before LumiNUS upload, as indicated in the table below. Copy your respective "Archive Naming", and then replace the "xxxxxxxxx" with your student ID number. Do not change any other part of the naming, except the "xxxxxxxxx".

Submission example: L1_Fri_AM_Alan Turing_Archive_A0131086Z

Name	Archive Naming [Note: L1 stands for lab 1, and it will be different for other labs]
A AKIL AHAMED	L1_Fri_AM_A AKIL AHAMED_Archive_XXXXXXXXX
Abdul Hadi Bin Abdul Samad	L1_Fri_AM_Abdul Hadi Bin Abdul_Archive_XXXXXXXXX
Adabelle Lim Ru Leng	L1_Fri_AM_Adabelle Lim Ru Leng_Archive_XXXXXXXXX
Alfred Wrong Jia Qing	L1_Fri_AM_Alfred Wrong Jia Qin_Archive_XXXXXXXXX
Alvin Goh Jia Hao	L1_Fri_AM_Alvin Goh Jia Hao_Archive_XXXXXXXXX
Alvinci Merquita	L1_Wed_AM_Alvinci Merquita_Archive_XXXXXXXXX
ANG KENG SIANG	L1_Wed_AM_ANG KENG SIANG_Archive_XXXXXXXXX
Aryl Ng Shen Le	L1_Wed_AM_Aryl Ng Shen Le_Archive_XXXXXXXXX
Au Yuan Xian	L1_Fri_AM_Au Yuan Xian_Archive_XXXXXXXXX
Bai Xiaoru	L1_Fri_AM_Bai Xiaoru_Archive_XXXXXXXXX
Bryan Yu Cheng You	L1_Fri_AM_Bryan Yu Cheng You_Archive_XXXXXXXXX
Chai Wei Lynthia	L1_Fri_AM_Chai Wei Lynthia_Archive_XXXXXXXXX
Cheang Zhi Yi Jordan	L1_Fri_AM_Cheang Zhi Yi Jordan_Archive_XXXXXXXXX
Chee Poh Hock	L1_Wed_AM_Chee Poh Hock_Archive_XXXXXXXXX
Cheng Wei Qiao	L1_Fri_AM_Cheng Wei Qiao_Archive_XXXXXXXXX
Cheung Po Rui Bryan	L1_Fri_AM_Cheung Po Rui Bryan_Archive_XXXXXXXXX
CHONG LEE TENG VALENCIA	L1_Wed_AM_CHONG LEE TENG VALEN_Archive_XXXXXXXXX
Davian Chan Sze Peng	L1_Fri_AM_Davian Chan Sze Peng_Archive_XXXXXXXXX
David Michael Woodside	L1_Fri_AM_David Michael Woodsi_Archive_XXXXXXXXX
ELJER CHUA	L1_Fri_AM_ELJER CHUA_Archive_XXXXXXXXX
FANG XINJIA	L1_Fri_AM_FANG XINJIA_Archive_XXXXXXXXX
Fidel Tan Yan Sheng	L1_Fri_AM_Fidel Tan Yan Sheng_Archive_XXXXXXXXX
Foo Fang Kiang	L1_Fri_AM_Foo Fang Kiang_Archive_XXXXXXXXX
Gao Zhixuan	L1_Fri_AM_Gao Zhixuan_Archive_XXXXXXXXX
Giam Xiong Yao	L1_Fri_AM_Giam Xiong Yao_Archive_XXXXXXXXX
Gillian Ho Xin Ying	L1_Fri_AM_Gillian Ho Xin Ying_Archive_XXXXXXXXX
Goh Jia Hong Edwin	L1_Wed_AM_Goh Jia Hong Edwin_Archive_XXXXXXXXX
Guinne Teresa Sng Yu Lin	L1_Fri_AM_Guinne Teresa Sng Yu_Archive_XXXXXXXXX
Hariharan Hadrian S/O Subramaniam	L1_Fri_AM_Hariharan Hadrian S._Archive_XXXXXXXXX
HO MING JUN	L1_Wed_AM_HO MING JUN_Archive_XXXXXXXXX
Ho Yi Shu Keon	L1_Wed_AM_Ho Yi Shu Keon_Archive_XXXXXXXXX
Hou Yinjiayi	L1_Fri_AM_Hou Yinjiayi_Archive_XXXXXXXXX
Ian Isaiah Tan Jun Wei	L1_Fri_AM_Ian Isaiah Tan Jun W_Archive_XXXXXXXXX
Jacob Zhang Zhiqiang	L1_Wed_AM_Jacob Zhang Zhiqiang_Archive_XXXXXXXXX
JEROME TEO SZE YONG	L1_Wed_AM_JEROME TEO SZE YONG_Archive_XXXXXXXXX
Jonathan Ang Xu Wen	L1_Wed_AM_Jonathan Ang Xu Wen_Archive_XXXXXXXXX
JONATHAN KHOO TENG YANG	L1_Fri_AM_JONATHAN KHOO TENG Y_Archive_XXXXXXXXX
Kabeta Takuma	L1_Wed_AM_Kabeta Takuma_Archive_XXXXXXXXX
Khoo Wu Jian Samuel	L1_Wed_AM_Khoo Wu Jian Samuel_Archive_XXXXXXXXX
KIM JOOHWAN	L1_Fri_AM_KIM JOOHWAN_Archive_XXXXXXXXX
Lau Wai Kit	L1_Wed_AM_Lau Wai Kit_Archive_XXXXXXXXX
LEE KE HUI	L1_Wed_AM_LEE KE HUI_Archive_XXXXXXXXX
Lee Shao Yu	L1_Wed_AM_Lee Shao Yu_Archive_XXXXXXXXX
Lek Ju Ying	L1_Wed_AM_Lek Ju Ying_Archive_XXXXXXXXX
Leong Ka Weng, Rachelle	L1_Fri_AM_Leong Ka Weng, Rache_Archive_XXXXXXXXX
LEW POH CHEN, DOUGLAS	L1_Fri_AM_LEW POH CHEN, DOUGLA_Archive_XXXXXXXXX
Long Deng Jie	L1_Wed_AM_Long Deng Jie_Archive_XXXXXXXXX
Markus Lim Yi Qin	L1_Wed_AM_Markus Lim Yi Qin_Archive_XXXXXXXXX
Mohamad Adam Bin Mohamad Yazid	L1_Wed_AM_Mohamad Adam Bin Moh_Archive_XXXXXXXXX
Muhammad Irfan Bin Zakaria	L1_Wed_AM_Muhammad Irfan Bin Z_Archive_XXXXXXXXX
Myat Thwe Naing	L1_Wed_AM_Myat Thwe Naing_Archive_XXXXXXXXX
Ng Etek	L1_Wed_AM_Ng Etek_Archive_XXXXXXXXX
Noorhakim Bin Jasman	L1_Wed_AM_Noorhakim Bin Jasman_Archive_XXXXXXXXX
NUR SYADIYAH BTE LUTFI	L1_Fri_AM_NUR SYADIYAH BTE LUT_Archive_XXXXXXXXX
ONG WEI SHENG	L1_Fri_AM_ONG WEI SHENG_Archive_XXXXXXXXX
PANG JUN WEN, ADRIC	L1_Fri_AM_PANG JUN WEN, ADRIC_Archive_XXXXXXXXX
PUN ZE YONG	L1_Wed_AM_PUN ZE YONG_Archive_XXXXXXXXX
Qiang Zhuang	L1_Wed_AM_Qiang Zhuang_Archive_XXXXXXXXX
QIU YI WEN	L1_Fri_AM_QIU YI WEN_Archive_XXXXXXXXX
R M RAAJAMANI	L1_Fri_AM_R M RAAJAMANI_Archive_XXXXXXXXX
RIZAVUR RAHMAN FASLUR RAHMAN	L1_Fri_AM_RIZAVUR RAHMAN FASLU_Archive_XXXXXXXXX
Ryan Tan Jun Hao	L1_Wed_AM_Ryan Tan Jun Hao_Archive_XXXXXXXXX
Saw Wee Kiat	L1_Wed_AM_Saw Wee Kiat_Archive_XXXXXXXXX
SIM BOWEN	L1_Fri_AM_SIM BOWEN_Archive_XXXXXXXXX
TAM LI NA	L1_Fri_AM_TAM LI NA_Archive_XXXXXXXXX
Tan Javen	L1_Wed_AM_Tan Javen_Archive_XXXXXXXXX
Tan Kai Hao Andrew	L1_Wed_AM_Tan Kai Hao Andrew_Archive_XXXXXXXXX
Tan Suet Ying	L1_Fri_AM_Tan Suet Ying_Archive_XXXXXXXXX
Tan Yeung Ming Sean Eugene	L1_Wed_AM_Tan Yeung Ming Sean _Archive_XXXXXXXXX
Teoh Yi Zheng	L1_Fri_AM_Teoh Yi Zheng_Archive_XXXXXXXXX
Tey Zi Le	L1_Wed_AM_Tey Zi Le_Archive_XXXXXXXXX
Thet Ke Min, Sonia	L1_Wed_AM_Thet Ke Min, Sonia_Archive_XXXXXXXXX
Trina Wern Qin Rong	L1_Fri_AM_Trina Wern Qin Rong_Archive_XXXXXXXXX
Venessa Chee Li Lin	L1_Fri_AM_Venessa Chee Li Lin_Archive_XXXXXXXXX
WANG HUA CHEN	L1_Fri_AM_WANG HUA CHEN_Archive_XXXXXXXXX
Wee Cheng Yuan Andrew	L1_Wed_AM_Wee Cheng Yuan Andre_Archive_XXXXXXXXX
Wee Xin Ze	L1_Wed_AM_Wee Xin Ze_Archive_XXXXXXXXX
Yeo Zhong Kang Dennis	L1_Wed_AM_Yeo Zhong Kang Denni_Archive_XXXXXXXXX
ZHONG SHUHAO	L1_Fri_AM_ZHONG SHUHAO_Archive_XXXXXXXXX