EE2026: DIGITAL DESIGN

Academic Year 2021-2022, Semester 2

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 **strongly 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: WEDNESDAY 26th JANUARY 2022, NOON]:

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.

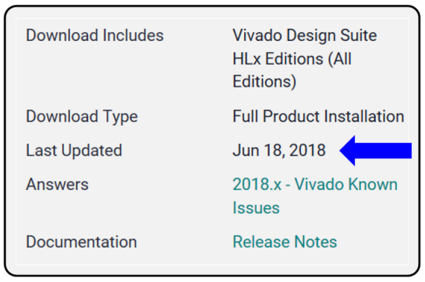
Software Weblink

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.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.*



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**:

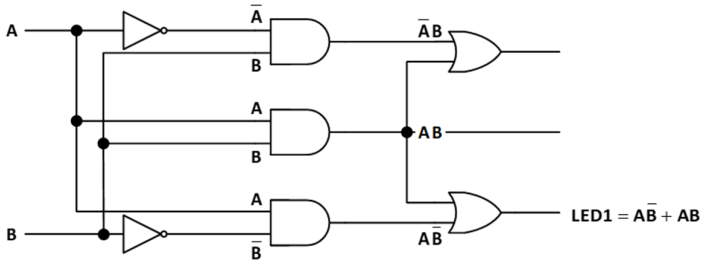
$$\mathbf{LED1} = A\bar{B} + AB$$

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 does not have any spaces or special characters, and that your **Project name** does not start with a number

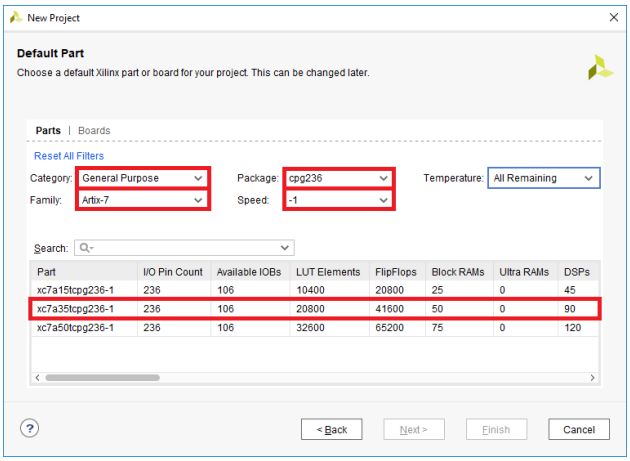
Project Type: Select **RTL Project**, and uncheck "**Do not specify sources at this time**"

Add sources:

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

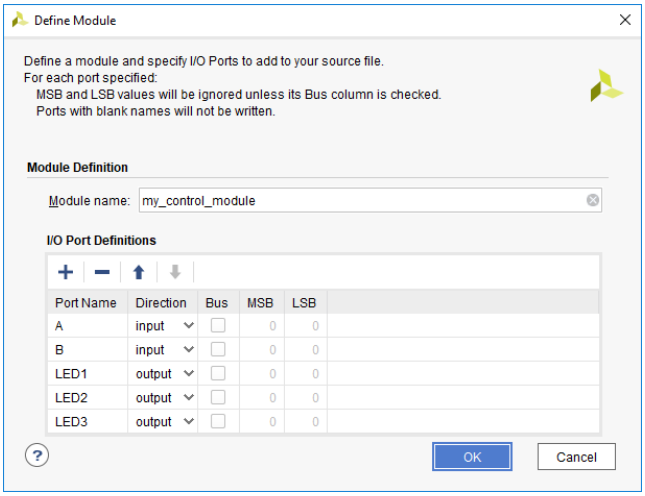
Add constraints (optional): Click on next without any changes

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



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.



Using Vivado Text Editor to Write Verilog HDL Code

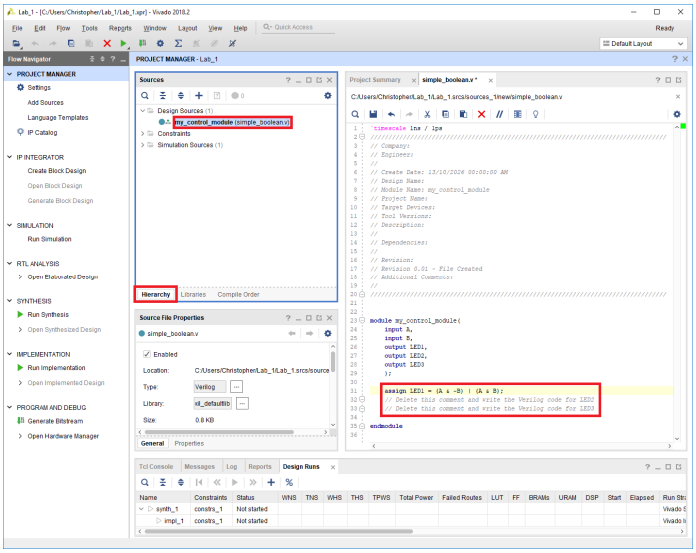
Open the module that has been created by double clicking on it in the Sources window

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	$\mathbf{\overline{A}}$	~
XOR	$\mathbf{A \oplus B}$	^



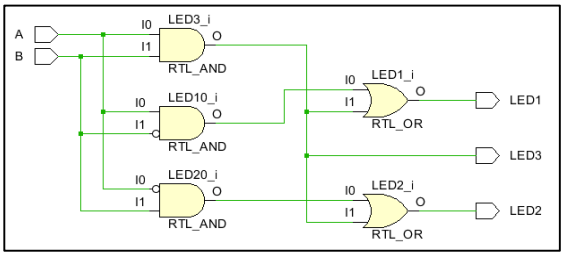
The **assign** statement causes the left hand side of the expression to be updated *every* time there is a change on the right hand side of the expression. It is therefore called a *continuous assignment* statement, describing combinational logic whereby the output on the left is a function of current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

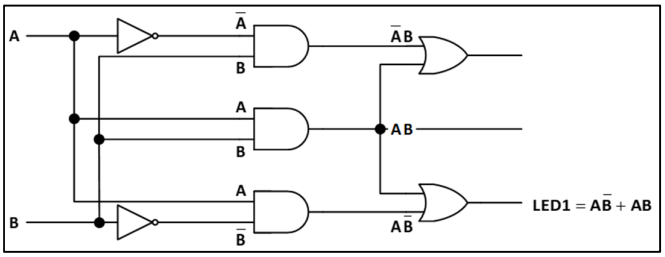
The statements on line 31 till line 33 execute concurrently. This is in contrast to sequential execution of statements in a computer programming language such as C, or procedural assignment that will be taught in subsequent lab sessions.

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?



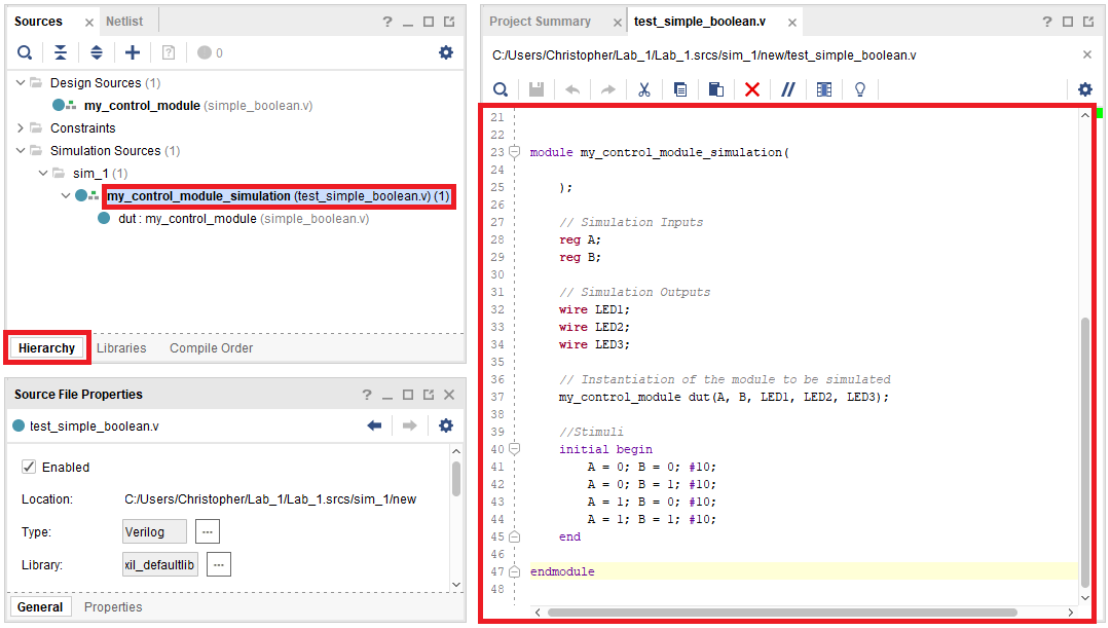


Testbench and Behavioural Simulation

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

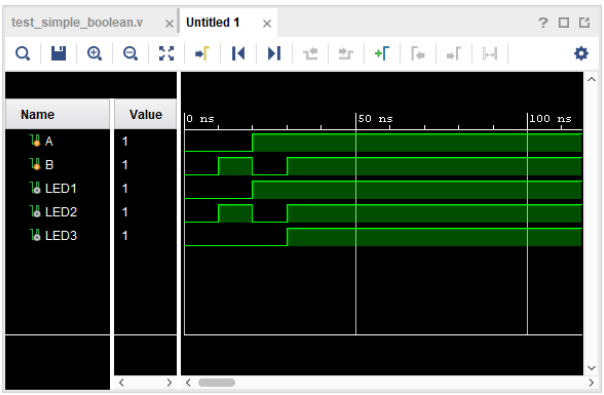
From the Sources window, open the simulation file. Then, within the simulation module, provide the following codes and save them, with the final screenshot looking similar to the image shown below:



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.



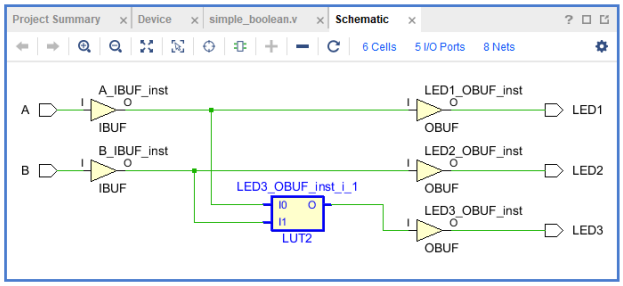
Synthesis

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.

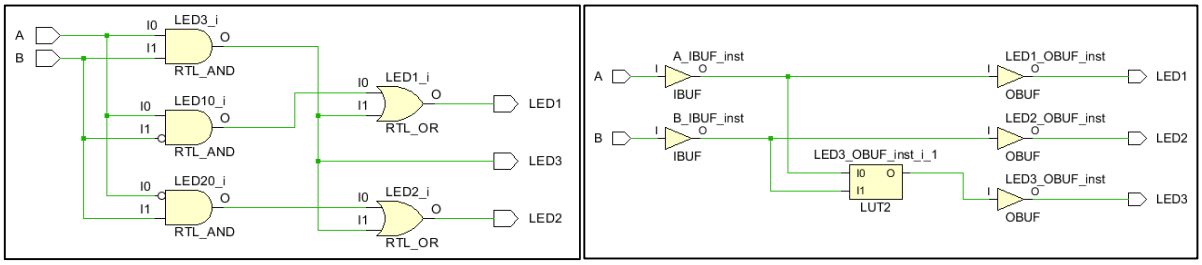
After the synthesis has been successfully completed, in the **Flow Navigator** window, under **SYNTHESIS**, expand **Open Synthesized Design**, and select **Schematics**. The schematic of the synthesised design will be generated and this synthesised circuit is an optimised version of the RTL schematic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

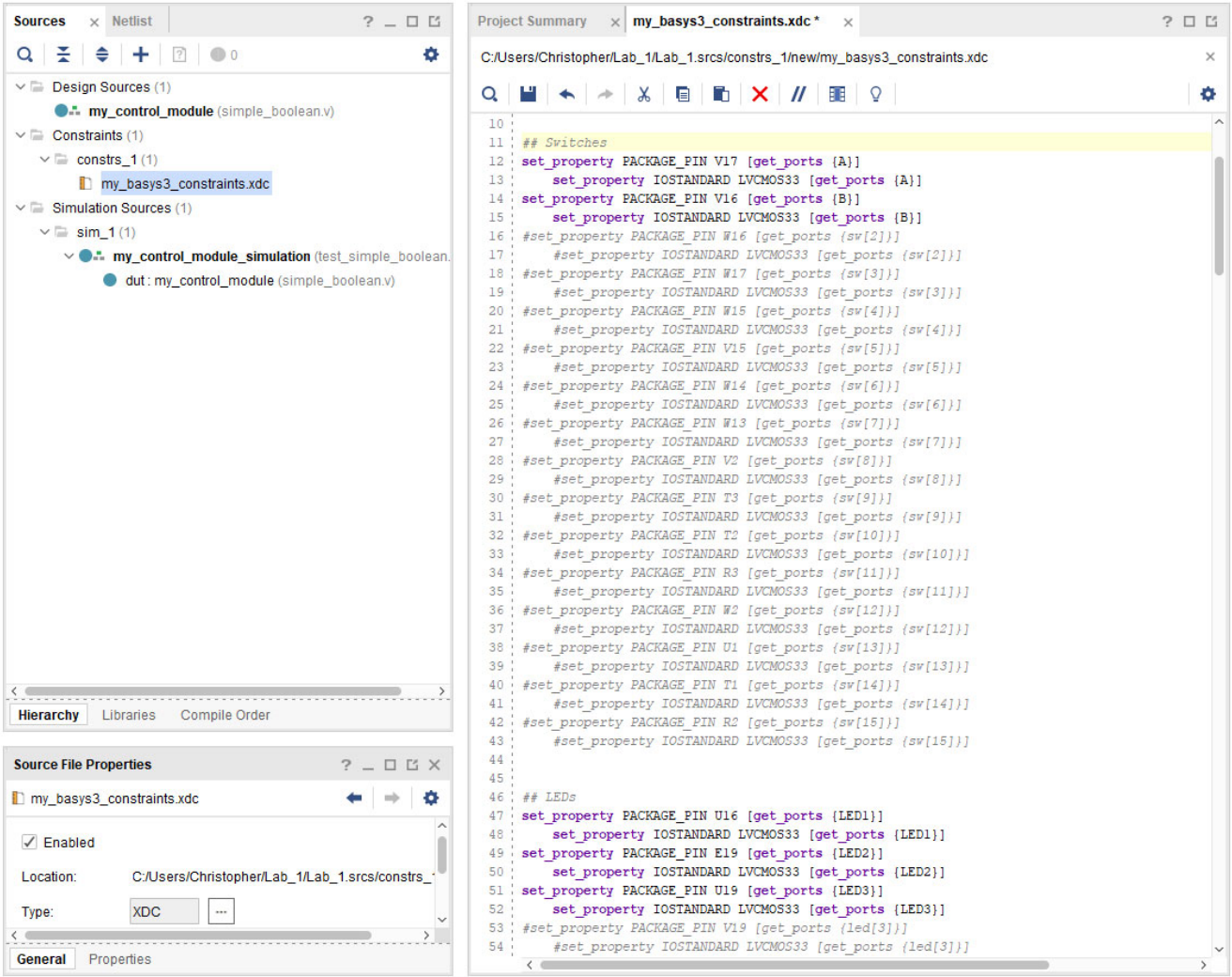


Design Constraints

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:



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.

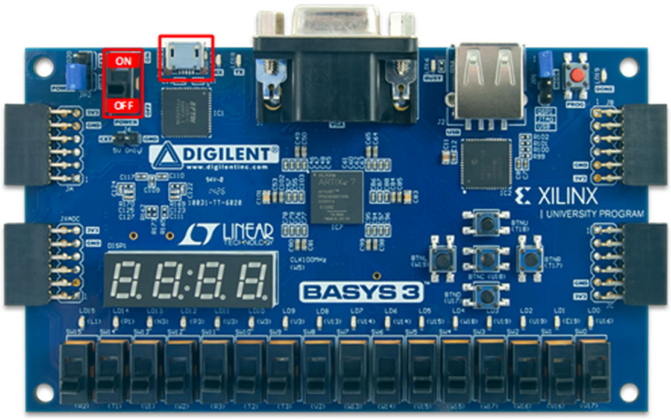
Before using your Basys 3 development board, and to prevent potential damage to it, take note of the following recommendations and warnings to extend the longevity of the device:

⚠ 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

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 1st rightmost numerical value of your student matriculation number (Subtask A)
- The five rightmost numerical values of your student matriculation number (Subtask B)
- The 2nd rightmost numerical value of your student matriculation number (Subtask C)
- The rightmost alphabet of your student matriculation number (Subtask C)

INITIALISATION

When the program starts, all 16 active-high switches (SW0 to SW15) are in the OFF position. All 16 active-high LEDs (LD0 to LD15) are also OFF. The seven segment displays must show the following patterns **exactly**, based on the **1st rightmost numerical value of your student matriculation number**:

1st Rightmost Numerical Value	0	1	2	3	4
Required 7-Segments Displays					

1st Rightmost Numerical Value	5	6	7	8	9
Required 7-Segments Displays					

SUBTASK A

Consider the 10 (ten) switches SW0 to SW9. Whenever any of these 10 switches are ON, the corresponding LED LD**X**, where **X** is a number ranging from 0 to 9, must be ON. Examples:

- If SW0 is ON, then LD0 must be ON
- If SW3, SW7 and SW9 are ON, then LD3, LD7 and LD9 must be ON

SW0 to SW9, and LD0 to LD9, must all be constraint.
SW10 to SW15, and LD10 to LD14, must be ignored (Do not put a constraint to switches SW10 to SW15 and LEDs LD10 to LD14).
LD15 requires constraint from SUBTASK B onwards.

SUBTASK B

Continuing from SUBTASK A, create your personal password based on the **five rightmost numerical values of your student matriculation number** (Ignore the alphabet character).

These five digits (May be less than five digits if you have duplicate numbers) will represent the switches that need to be ON, while all the other switches between SW0 to SW9 must be OFF, to be considered a correct password.

If the password entered by the user is the correct password, then LED LD15 must turn ON. LD15 is OFF whenever the password is incorrect.

SUBTASK C

When the password from SUBTASK B is correct and LD15 is ON, it is also required to display the rightmost alphabet of your student matriculation number on some specific anodes of the 7-segment displays. The character must be displayed **exactly** as indicated:

Rightmost Alphabet	A	B	E	H	J	L	M	N	R	U	W	X	Y
Required 7-Segments Character	A	b	E	H	J	L	n	n	r	U	U	H	y

The anode on which the character should be displayed is dependent on the **2nd rightmost numerical value** of your student matriculation number, as indicated in the table below:

2nd Rightmost Numerical Value	Anode AN3	Anode AN2	Anode AN1	Anode AN0
0	**ON**	**ON**	OFF	**ON**
1	**ON**	**ON**	OFF	OFF
2	**ON**	OFF	**ON**	**ON**
3	**ON**	OFF	**ON**	OFF
4	**ON**	OFF	OFF	**ON**
5	**ON**	OFF	OFF	OFF
6	OFF	**ON**	**ON**	**ON**
7	OFF	**ON**	**ON**	OFF
8	OFF	**ON**	OFF	**ON**
9	OFF	**ON**	OFF	OFF

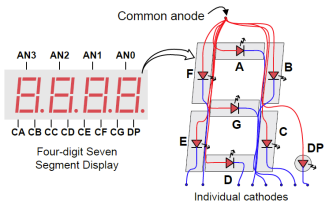
When the password from SUBTASK B is not correct, it is compulsory for the seven-segment displays to show the set of characters indicated in the INITIALISATION phase.

SUGGESTIONS

- 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, as this is meant to be a warming-up assignment
- The following will be taught in subsequent lectures / tutorials / labs, and are thus **not necessary** in lab 1:
    - if-else functions
    - always blocks
    - multi-bits vector

GETTING STARTED WITH THE SEVEN-SEGMENT DISPLAYS

There are 7 LED segments in each display, with an additional decimal point segment. 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)

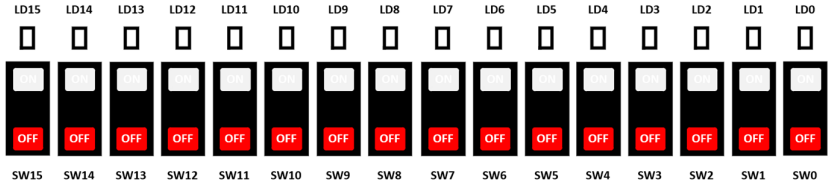
In your constraint file, it is compulsory to put constraints to the 8 segments (7 segments + decimal point) of the seven-segment display, and to the 4 anodes of the seven-segment display.

EXAMPLE:

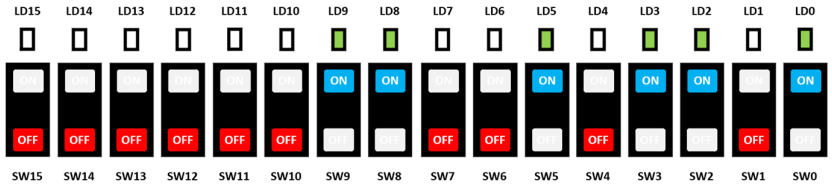
If your student matriculation number is A0159089Y, then:

1st rightmost numerical value:	9
Five rightmost numerical values:	59089
2nd rightmost numerical value:	8
Rightmost alphabet:	Y

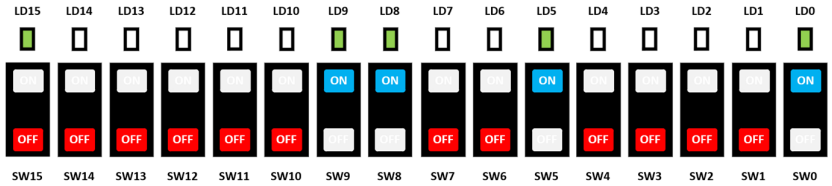
INITIALISATION



WRONG PASSWORD



CORRECT PASSWORD



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

- Upload to LumiNUS EE2026 -> Files -> Lab and Project - Materials and Submissions -> Lab 1 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 26th January 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 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 (COMPULSORY renaming before just LumiNUS upload):

It is **compulsory to rename your project archive**, 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.

Name	Archive Naming
Aaron Chan Zhi	L1_Thurs_AM_Aaron Chan Zhi_476_Archive
Ajay Shanker	L1_Thurs_AM_Ajay Shanker_806_Archive
Alphonsus Teow Rui Jie	L1_Thurs_AM_Alphonsus Teow Rui Jie_502_Archive
Alvin Ben Abraham	L1_Thurs_AM_Alvin Ben Abraham_394_Archive
Amadeus Lim Ding Shin	L1_Thurs_AM_Amadeus Lim Ding Shin_412_Archive
Amit Rahman	L1_Thurs_AM_Amit Rahman_599_Archive
Ang Jia Le Marcus	L1_Thurs_AM_Ang Jia Le Marcus_025_Archive
Chan Ee Hong	L1_Thurs_AM_Chan Ee Hong_898_Archive
Chen Zi Han	L1_Thurs_AM_Chen Zi Han_549_Archive
Chien Jing Wei	L1_Thurs_AM_Chien Jing Wei_540_Archive
CHUA WEI XUAN	L1_Thurs_AM_CHUA WEI XUAN_716_Archive
Chua Wen Xin Kyrene	L1_Thurs_AM_Chua Wen Xin Kyrene_431_Archive
Darren Loh Rui Jie	L1_Thurs_AM_Darren Loh Rui Jie_289_Archive
Dennis Wong Guan Ming	L1_Thurs_AM_Dennis Wong Guan Ming_806_Archive
Huang Yu Chiao	L1_Thurs_AM_Huang Yu Chiao_102_Archive
Ivan Theng Wen Rong	L1_Thurs_AM_Ivan Theng Wen Rong_344_Archive
Jia Yixuan	L1_Thurs_AM_Jia Yixuan_150_Archive
Karthikeyan Vigneshram	L1_Thurs_AM_Karthikeyan Vigneshram_697_Archive
Leong Wei Lun, Alfred	L1_Thurs_AM_Leong Wei Lun Alfred_609_Archive
Liu Junhao	L1_Thurs_AM_Liu Junhao_523_Archive
Marvin Pranajaya	L1_Thurs_AM_Marvin Pranajaya_683_Archive
Ng Sihan, Ian	L1_Thurs_AM_Ng Sihan Ian_817_Archive
Ong Zhi Hong	L1_Thurs_AM_Ong Zhi Hong_922_Archive
Raymond Bala	L1_Thurs_AM_Raymond Bala_127_Archive
See Zhuo Rui Jorelle	L1_Thurs_AM_See Zhuo Rui Jorelle_490_Archive
Shanmugam Surya	L1_Thurs_AM_Shanmugam Surya_189_Archive
Shawn Tan Jinhui	L1_Thurs_AM_Shawn Tan Jinhui_247_Archive
Shin Donghun	L1_Thurs_AM_Shin Donghun_808_Archive
Stefan Choo Bin Hao	L1_Thurs_AM_Stefan Choo Bin Hao_098_Archive
Syed Muhamad Amali B Syed A A	L1_Thurs_AM_Syed Muhamad Amali B Syed_373_Archive
Teh Zi-Chun	L1_Thurs_AM_Teh ZiChun_328_Archive
Wen Chen Yu	L1_Thurs_AM_Wen Chen Yu_109_Archive
Wilson Ng Jing An	L1_Thurs_AM_Wilson Ng Jing An_686_Archive
Yong Chin Han	L1_Thurs_AM_Yong Chin Han_814_Archive
Yuan Xinrui	L1_Thurs_AM_Yuan Xinrui_211_Archive