EE2026: DIGITAL DESIGN

Academic Year 2022-2023, Semester 2

LAB 3: Sequential Circuits in Verilog

OVERVIEW

A sequential circuit is one where the outputs depend on the current inputs and the sequence of past inputs. As a result, a sequential circuit has memory, also called states. In this lab, some basic sequential circuits will be designed to make an LED blink at various speeds.

The pre-requisites for this lab are:

- A very good understanding and application of dataflow modelling and structural modelling in designing modules.
- Knowing how to use the Vivado IDE well.
- Familiarity and knowledge on how to use "*Set as Top*", "*reg*" and "*wire*".

This lab will cover the following:

- Using a signal that inverts itself periodically, which shall be called **CLOCK**.
- Making a physical LED blink by using the Basys 3 development board clock signal.

Tasks for this lab include:

- Creating a slower clock from a faster clock.
- Having a physical LED blink noticeably on the Basys 3 development board, by using the slower clock.
- Using a switch to make a physical LED blink at two different speeds on the Basys 3 development board.

GRADED ASSIGNMENT [CANVAS SUBMISSION: WEDNESDAY 22nd FEBRUARY 2023, 10 P.M.]:

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

THE BLINKING LED

A simple blinking LED is required to be implemented on the FPGA. To do this, a new signal, **CLOCK**, will be introduced.

The **CLOCK** signal is an external input signal that resembles a square wave of 50% duty cycle. If this **CLOCK** signal is connected directly to a physical LED, the latter will light up when the signal is HIGH, and will switch off when the signal is LOW, as illustrated in **Figure 3.2**.

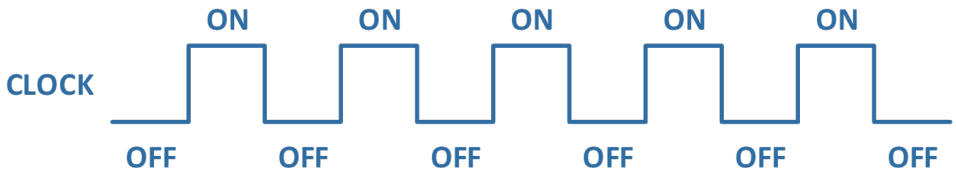


Figure 3.2: *A **CLOCK** signal with 50% duty cycle*

A simple dataflow description in Verilog for a blinky module is written first, followed by a simulation source to verify the design. To create the square wave, or **CLOCK** signal, in the simulation source, a new section of codes will now be introduced:

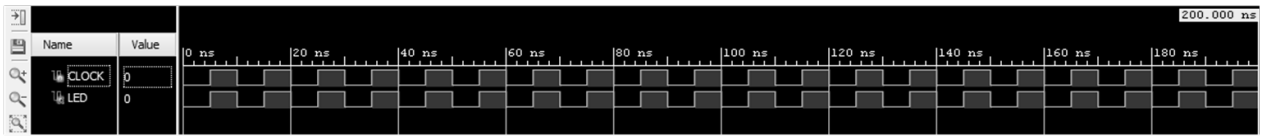
Verilog code for blinky, using the dataflow method

```
module blinky (input CLOCK, output LED);

    assign LED = CLOCK;

endmodule
```

Simulation source code to test the blinky design

```
`timescale 1ns / 1ps

module test_blinky( );

    reg CLOCK; wire LED;

    blinky dut (CLOCK, LED);

    initial begin
        CLOCK = 0;
    end

   always begin
       #5 CLOCK = ~CLOCK;
   end

endmodule
```

Expected simulation waveform for the blinky design



UNDERSTANDING | TASK 2

Based on the Verilog code and simulation results, check your understanding by answering the following questions:

1. What is the unit of time being used in the simulation source?

   ns

2. Every 5 units of time, the value of **CLOCK** is being inverted. What is the clock frequency being used in this simulation?

   2 * 10^8

3. What would happen if the testbench code **CLOCK = 0** is removed?

   no value

For the hardware implementation, instead of using an external signal generator for the **CLOCK** signal to the Artix-7 FPGA, the Basys 3 development board includes a single 100 MHz clock generator connected to pin W5 of the Artix-7 FPGA.

Using the original contents of the ***Basys3_Master.xdc*** in your constraint file, follow these steps:

1. Uncomment lines 7 to 9 to create a clock signal of 100 MHz with 50% duty cycle. If required, rename the signal to the name used in your **blinky** code. In our example, the name **CLOCK** was used, and the final changes may look similar to **Figure 3.3**.

2. Configure the output signal **LED** that is present in your **blinky** code (or the name chosen by you while writing the code) by linking it to any physical LED on the Basys3 development board.

```
1 ## This file is a general .xdc for the Basys3 rev B board
2 ## To use it in a project:
3 ## - uncomment the lines corresponding to used pins
4 ## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project
5
6 ## Clock signal
7 set_property PACKAGE_PIN W5 [get_ports CLOCK]
8     set_property IOSTANDARD LVCMOS33 [get_ports CLOCK]
9     create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports CLOCK]
10
11 ## Switches
12 #set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
13     #set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
14 #set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
15     #set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
```

Figure 3.1: *Modifying your constraint file, based on the contents of the Basys3_Master.xdc*

UNDERSTANDING | TASK 3

You may optionally generate the bitstream and upload your code to the Basys3 development board. What do you "*notice*" about the "*blinking*" LED?

_____

_____

COUNTER FOR A SLOWER CLOCK

To be able to observe a blinking LED at a frequency that is visible to the human eyes, modifications need to be done to the Verilog code to have a slower clock. Let us introduce a temporary variable **COUNT** that is incremented by 1 at every rising edge (transition from low to high; also called a positive edge) of the **CLOCK** signal, as shown in **Figure 3.4**.

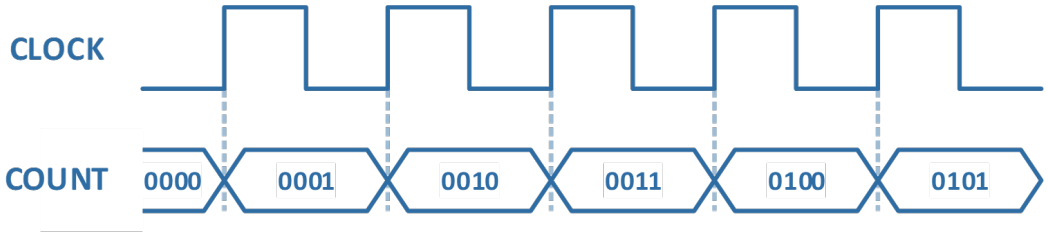


Figure 3.4: *Increasing* ***COUNT*** *at each rising edge of* ***CLOCK***

Verilog code for an incrementing counter of 4-bit, using behavioural modelling

```
module slow_blinky_module (input CLOCK);

    reg [3:0] COUNT = 4'b0000;

    always @ (posedge CLOCK) begin
        COUNT <= COUNT + 1;
    end

endmodule
```

Now, use a conditional statement that changes the state of an output signal, called SLOW_CLOCK, whenever the COUNT reaches zero. The initial state of SLOW_CLOCK can be zero or one. Here, since the 4-bits COUNT has 16 possible states, it would mean that SLOW_CLOCK changes state after 16 clock cycles of CLOCK. In other words, SLOW_CLOCK is 1/16th the clock speed of CLOCK.

Partial Verilog code for toggling SLOW_CLOCK signal, using behavioural modelling

```
    always @ (posedge CLOCK) begin
        COUNT <= COUNT + 1;
        SLOW_CLOCK <= ( COUNT == 4'b0000 ) ? ~SLOW_CLOCK : SLOW_CLOCK ;
    end
```

UNDERSTANDING | TASK 4

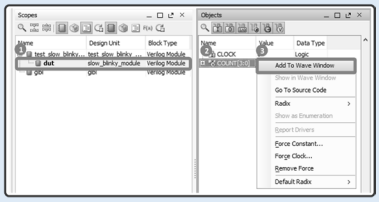
Simulate the Verilog codes, so that a waveform that toggles every 16th clock cycle can be observed

Finally, explore what happens if SLOW_CLOCK is forcefully toggled before the overflow to 0 occurs. To do so, add a conditional statement to allow SLOW_CLOCK to toggle at earlier clock cycle instead of waiting for 16 clock cycles. Simulate the Verilog codes again and observe if the waveforms have toggled earlier.

FURTHER ANALYSIS IN THE SIMULATION WAVEFORM WINDOW

By default, the simulation window only shows the waveforms of input and output signals. To see the waveforms of variables during the simulation, such as the variable **COUNT**:

1. Select the **dut** under the simulation module being used
2. In the **Objects** window, right click on the **COUNT** variable
3. Choose **Add To Wave Window**



After adding the variable **COUNT** to the wave window, the current simulation needs to be re-run. Follow these steps:

1. Restart the simulation
2. Set the simulation time and units
3. Run the simulation for the amount of time set in step 2

This is an important step to remember if you want to simulate for more than the default 1000 ns of simulation!!!

The **COUNT** variable can then be expanded by clicking on the **+** symbol to the left of **COUNT**. This allows for every individual bit to be observed as independent waveforms:

UNDERSTANDING | TASK 5

Create 4 different clock signals with approximately (A precise value is not required) the following clock frequencies:

- clock_A : 1000 Hz
- clock_B : 100 Hz
- clock_C : 10 Hz
- clock_D : 1 Hz

Simulate clock_A to confirm that you have a clock of approximately 1000 Hz.

Simulate clock_B to confirm that you have a clock of approximately 100 Hz.

Self-thinking: Why do you think it is not always feasible to simulate the very slow clocks, such as 10 Hz and 1 Hz?

THE NOTICEABLE BLINKING LED (POST-LAB NON-GRADED – NO SUBMISSION REQUIRED):

It is strongly encouraged to complete this practice task before working on the graded post-lab assignment!

Assume LD0 on the Basys 3 development is the LED to blink.

Consider the three switches SW0, SW1 and SW2, and their behaviours:

- When these four switches are OFF, LD0 must be OFF
- When SW0 is ON, LD0 must blink at 100 Hz
- When SW1 is ON, LD0 must blink at 10 Hz
- When SW2 is ON, LD0 must blink at 1 Hz

Implement the above requirement on the Basys 3 development board. You can use the multiplexers, if-else statements, or case statements to write the conditional codes. It is also up to you to decide which switch has higher importance.

Note: For the EE2026 labs, a blinking frequency of 1 Hz means that the LED should be ON for 0.5 seconds, and OFF for around 0.5 seconds, for each blinking cycle

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 2nd rightmost numerical value of your student matriculation number (Subtask B)
- The five rightmost numerical values of your student matriculation number (Subtask D)
- The rightmost alphabet of your student matriculation number (Subtask D)

INITIALISATION

When the program starts, switches SW0 to SW15 must be in the OFF position. The seven segment displays are all OFF.

SUBTASK A

At the start of the program, a set of LEDs starting from LD0 to MAX_LED (Including) are required to turn ON after every TIME_COUNT. Other LEDs, from MAX_LED (Excluding) to LD15, are OFF. MAX_LED is dependent on the **1st rightmost numerical value of your student matriculation number**, as indicated in the table below:

1st Rightmost Numerical Value	MAX_LED	TIME_COUNT
0	LD14	0.20 seconds (Error of ± 0.05 seconds accepted)
1	LD13	0.36 seconds (Error of ± 0.05 seconds accepted)
2	LD12	0.54 seconds (Error of ± 0.05 seconds accepted)
3	LD11	0.75 seconds (Error of ± 0.05 seconds accepted)
4	LD10	1.00 seconds (Error of ± 0.05 seconds accepted)
5	LD9	1.20 seconds (Error of ± 0.05 seconds accepted)
6	LD8	1.11 seconds (Error of ± 0.10 seconds accepted)
7	LD7	1.00 seconds (Error of ± 0.10 seconds accepted)
8	LD6	0.86 seconds (Error of ± 0.10 seconds accepted)
9	LD5	0.67 seconds (Error of ± 0.10 seconds accepted)

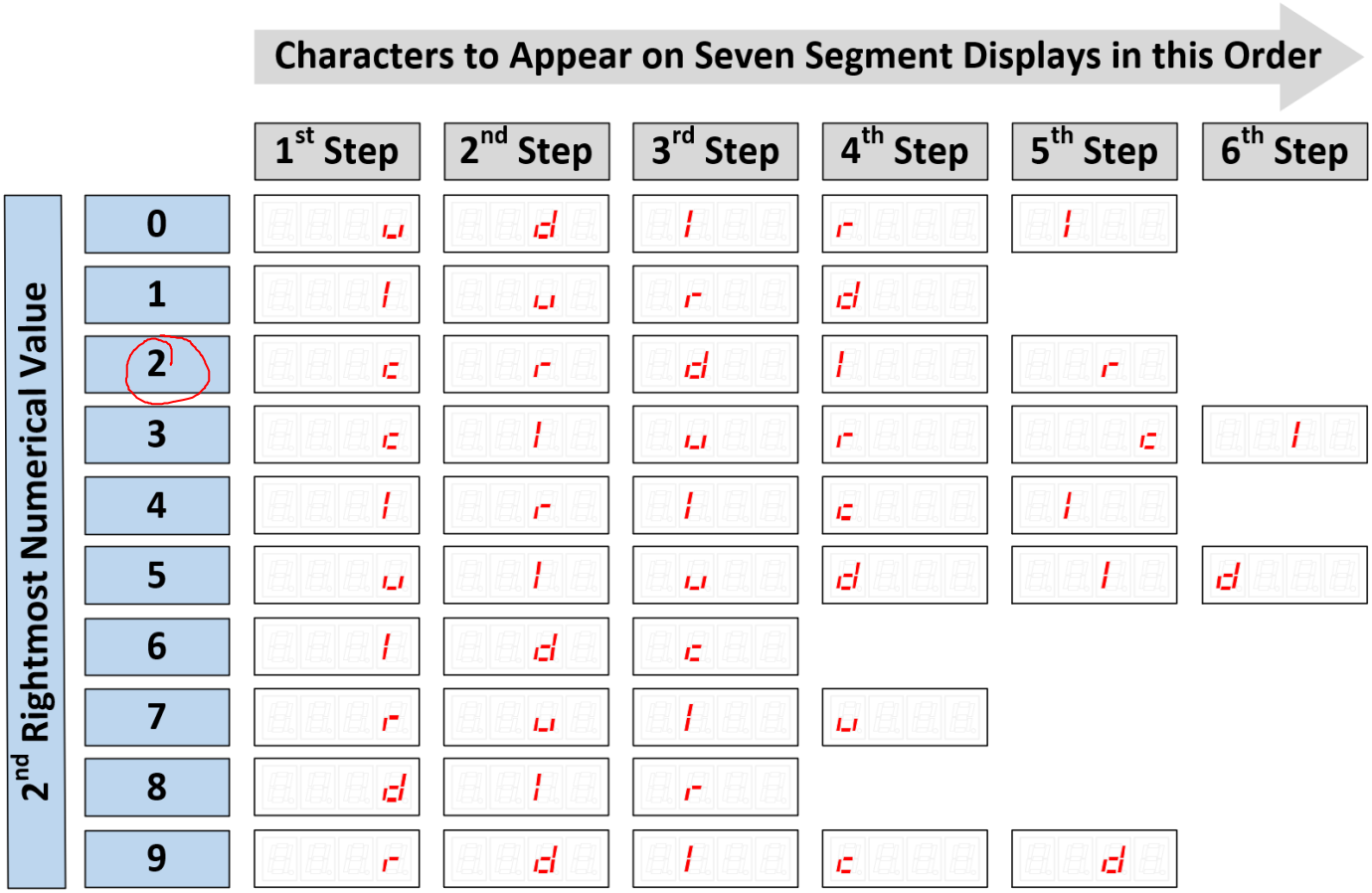
When all the LEDs from LD0 to MAX_LED are ON, the user can turn ON certain switches. These switches should make certain LEDs blink, as indicated in the truth table below (X is the don't care condition):

INPUTS			OUTPUTS			
SW2	**SW1**	**SW0**	**MAX_LED to LD2**	**LD2**	**LD1**	**LD0**
0	0	0	ON	ON	ON	ON
0	0	1	ON	ON	ON	Blink at 1 Hz
0	1	X	ON	ON	Blink at 10 Hz	ON
1	X	X	ON	Blink at 100 Hz	ON	ON

(Error of ± 10% accepted for all blinking frequencies)

SUBTASK B

When MAX_LED is ON at the end of SUBTASK A, the user will then be required to press certain pushbuttons based on the characters that appear on the seven segment displays. Recall that the seven segment displays do not show anything before MAX_LED is ON.



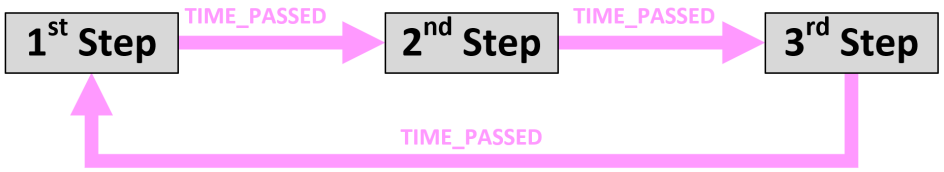
u	UP Pushbutton (BTNU) must be pressed before next step
r	RIGHT Pushbutton (BTNR) must be pressed before next step
d	DOWN Pushbutton (BTND) must be pressed before next step
l	LEFT Pushbutton (BTNL) must be pressed before next step
c	CENTER Pushbutton (BTNC) must be pressed before next step

After the user has pressed the required pushbuttons in the correct order, the system goes into an "UNLOCKED" mode, which is indicated by LD15 being ON. When LD15 is ON, the seven segment displays must show the 1st step again before proceeding to SUBTASK C

SUBTASK C

While in "UNLOCKED" mode, it is required to cycle sequentially through the first three steps at a speed determined by switches SW2, SW1, and SW0. It takes TIME_PASSED amount of time to move to the next step. After the 3rd Step has passed, it is required to restart from the 1st Step, and the process of cycling through the three steps continues infinitely. The value for TIME_PASSED, as well as an illustration of the sequential cycle, is shown below:

SW2	SW1	SW0	TIME_PASSED
0	0	0	*Infinity* (Character paused / No change in characters)
0	0	1	0.500 seconds (Error of ± 0.100 seconds accepted)
0	1	X	0.050 seconds (Error of ± 0.010 seconds accepted)
1	X	X	0.005 seconds (Error of ± 0.001 seconds accepted)



Note: The blinking LEDs and other LEDs from SUBTASK A must still work while doing SUBTASK C

SUBTASK D

Anytime throughout the whole program (SUBTASK A, B and C), if the user turns ON SW15 for 3 seconds (Error of ± 0.3 seconds accepted), then the following two things must happen:

- All LEDs must turn OFF and stop blinking, except for the LEDs LD**X** that must be ON. **X** represents each one of the five rightmost numerical values of your student matriculation number. Example: If the five values are 59089, then LD0, LD5, LD8 and LD9 must be ON.

- The seven segment displays must show the last character of your student matriculation on all four anodes (AN3, AN2, AN1, AN0), with the characters as indicated below:

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

If the user turns SW15 OFF again, immediately (Unlike the 3 seconds waiting time when SW15 is turned ON) the LEDs LD0 to LD15, as well as the seven segment displays should work as described in all the previous SUBTASKS.

SUGGESTIONS

- A counter whose value can change at each clock cycle can be considered. For example, that counter can count 0, 1, 2, 3, 0, 1, 2, 3 etc ... Following that, if-else statements, multiplexers, or case statement that indicates what to do at each specific counter value can be created

  Case statements are recommended here, and the case statement template is given below.

```
case (expression)
    case_item: statement or statement_group
    case_item: statement or statement_group
    default: statement or statement_group
endcase
```

- An example in using case is given below.

```
always @ (posedge clk_25_mhz)
begin
    case (counter_value)
       2'd0:
           begin
                  my_value_a <= 20;
                  my_value_b <= 40;
           end
       2'd1:
           begin
                  my_value_a <= 100;
                  my_value_b <= 200;
           end
       2'd2: my_value_c <= 5;
       default: my_value_d <= 9;
    endcase
end
```

- Case statements and if-else statements must be inside an always block.

- Conditional statements using multiplexers can be within or outside an always block.

- **Be careful of parallel execution of the always blocks. Multi-driven nets indicate that there are conflicting values being given to the same signal from different always blocks.** For example, one cannot tell a signal to increase at a time instant t, and at that same time instant t, telling it to decrease

- Refer to **http://tiny.cc/ee2026wiki** for more details on commonly encountered errors.

LINK TO BASYS3 DEMONSTRATION EXAMPLE:

EE2026 - Assignment 3 Demonstration.m4v

CANVAS SUBMISSION INSTRUCTIONS

- Ensure that your bitstream has been successfully generated and tested on your Basys 3 development board **BEFORE** archiving your Vivado workspace for CANVAS 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 CANVAS EE2026 -> Assignments -> Lab 3 Graded Assignment -> Lab 3 Submission (On-Time)

- Download your CANVAS 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 CANVAS upload must be completed by **Wednesday 22nd February 2023, 10:00 P.M.** 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: CANVAS EE2026 -> Assignments -> Lab 3 Graded Assignment -> Lab 3 Submission (Late)

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 ***CANVAS 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 just before CANVAS upload):

It is **compulsory to rename your project archive** just before the CANVAS 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.

CANVAS may automatically add a suffix (Example. "-1", "-2", "-3" etc.) at the end of the filename if you submit multiple files. That is acceptable and we will grade the latest file submitted within that on-time folder.

Archive Naming
L3_Mon_AM_ARIF KHALID_983_Archive
L3_Wed_PM_ARTEMIS NGOH_342_Archive
L3_Mon_AM_AUNG PHONE NAING_363_Archive
L3_Wed_PM_BENJAMIN CHRISTOPHER TOH_702_Archive
L3_Mon_AM_BENJAMIN LONG WEI MING_953_Archive
L3_Wed_AM_BIAN RUI_671_Archive
L3_Wed_PM_BRANDON OWEN SJARIF_179_Archive
L3_Wed_PM_BUI DUC THANH_820_Archive
L3_Wed_AM_BUI PHUONG NAM_451_Archive
L3_Wed_PM_CALEB CHAN JIA LE_569_Archive
L3_Wed_PM_CHAO YIJU_863_Archive
L3_Wed_PM_CHEAH HAO YI_666_Archive
L3_Wed_AM_CHEN JUNHAN JERALD_262_Archive
L3_Mon_AM_CHENVISUWAT NITA_872_Archive
L3_Wed_AM_CHERAN LEE YENNING_991_Archive
L3_Wed_PM_CHEW JUN WEI MAX_381_Archive
L3_Mon_AM_CHEW SI NING KACEY_433_Archive
L3_Wed_AM_CHIA YU XUAN_884_Archive
L3_Wed_PM_CHIEW YI XIANG_756_Archive
L3_Wed_AM_CHONG YONG RUI_594_Archive
L3_Wed_PM_CHUA ZHONG HENG_908_Archive
L3_Mon_AM_CHUAH WAN JUIN_377_Archive
L3_Wed_PM_CLEON CHENG RUI FENG_796_Archive
L3_Mon_AM_CLEON LIEW GE WEI_788_Archive
L3_Wed_PM_DEEPANJALI DHAWAN_498_Archive
L3_Wed_PM_DENNIS ONG QINKANG_605_Archive
L3_Wed_PM_DEXTER HOON YONG EN_166_Archive
L3_Wed_PM_DIVAKARAN MANUSHRI_824_Archive
L3_Mon_AM_DYLAN CHIA TIAN_729_Archive
L3_Wed_AM_DYLAN HO SHU JIE_536_Archive
L3_Wed_AM_EE JIA EN JARED_035_Archive
L3_Wed_PM_ER JUN ZE_233_Archive
L3_Mon_AM_EU ZHENG XI_767_Archive
L3_Wed_PM_EUGENE ANG JIA SHING_189_Archive
L3_Mon_AM_FARIS HAMID SIRRAJ_265_Archive
L3_Wed_AM_FELICIA BEATRICE BUDIAWAN_855_Archive
L3_Mon_AM_FOO YANG WEI JEROME_012_Archive
L3_Mon_AM_GAO YUN FAN_187_Archive
L3_Mon_AM_GARLAPATI SAI CHAITANYA_949_Archive
L3_Mon_AM_GAU KIAT LOK JERALD_058_Archive
L3_Wed_PM_GOH JING HONG_722_Archive

L3_Mon_AM_GOH YI XUAN_250_Archive
L3_Wed_PM_GRACE ZHU XING YU_936_Archive
L3_Mon_AM_GUAN XIAO_837_Archive
L3_Mon_AM_HAMADA MASAHIRO_120_Archive
L3_Wed_PM_HANG TIAN_725_Archive
L3_Wed_PM_HO ZHAN RUI GLENN_662_Archive
L3_Wed_PM_HONG LIN SHANG_890_Archive
L3_Mon_AM_HOO TENG JUAN_494_Archive
L3_Mon_AM_HUI YU CONG_397_Archive
L3_Mon_AM_HUNG HIN WANG CLEMENT_615_Archive
L3_Wed_AM_JAVIENNE YEO MYN_441_Archive
L3_Mon_AM_JOHN TOH JIA JUN_603_Archive
L3_Wed_PM_KHAIRUL AIZAT B MD HALIM_315_Archive
L3_Mon_AM_KHOO YOU RUN_353_Archive
L3_Wed_PM_KIM TAE WON_392_Archive
L3_Wed_PM_KISHORE SO ASOKAN_812_Archive
L3_Wed_PM_KOH CHEE HENG_333_Archive
L3_Mon_AM_KOH JING JIE MARCUS_625_Archive
L3_Mon_AM_KOH NGIAP HIN_229_Archive
L3_Wed_AM_KWUA CHUN REN_262_Archive
L3_Wed_PM_LEE JUN HAO BRYAN_090_Archive
L3_Wed_PM_LEE JUN HUI ANSENN_865_Archive
L3_Wed_AM_LEE ZHI XUAN_363_Archive
L3_Wed_PM_LEE ZHI ZHONG MOSES_669_Archive
L3_Wed_PM_LEI HAO_291_Archive
L3_Wed_PM_LEONARDO ONG DINGCHAO_944_Archive
L3_Wed_AM_LEONG HOI MING JOSHUA_256_Archive
L3_Mon_AM_LEONG HUEN WENG_012_Archive
L3_Wed_AM_LEONG HUNG REY_590_Archive
L3_Wed_PM_LEONG SONG ZHU OWEN_408_Archive
L3_Wed_PM_LEONG YAT PANG_470_Archive
L3_Wed_PM_LI MINGYUAN_670_Archive
L3_Mon_AM_LIM HONG YAO_619_Archive
L3_Wed_AM_LIM ZHENG RONG_047_Archive
L3_Mon_AM_LINUS PUAH JIA HE_811_Archive
L3_Wed_AM_LIONG WEI YONG DEEN_930_Archive
L3_Mon_AM_LIU XIAOGE_254_Archive
L3_Mon_AM_LOH JOO HOE_456_Archive
L3_Mon_AM_LOH YUAN LONG KEDRIAN_084_Archive
L3_Mon_AM_LU BINGYUAN_325_Archive
L3_Mon_AM_MAN JUNCHENG_844_Archive
L3_Wed_PM_MANOJ DORAIRAJAN_508_Archive
L3_Wed_PM_MATTHEW LIU ZHEN JIE_439_Archive
L3_Wed_PM_MITCH MALVIN_911_Archive
L3_Wed_AM_MUSTAFA ANIS HUSSAIN_072_Archive
L3_Wed_PM_MUTHUKRISHNAN NAVYA_130_Archive
L3_Mon_AM_MUTHYA NARAYANACHARY AKHI_509_Archive
L3_Wed_PM_NAM SANGJUN_260_Archive
L3_Wed_PM_NAZRUL SYAHMI BIN MURAD_711_Archive
L3_Mon_AM_NG DE QI_171_Archive
L3_Mon_AM_NG KAI WEN_144_Archive

L3_Wed_PM_NG LIXUAN NIXON_667_Archive
L3_Mon_AM_NG YAN ZHEN_909_Archive
L3_Wed_PM_NGUYEN DUC THANG_361_Archive
L3_Wed_AM_NGUYEN QUANG ANH_912_Archive
L3_Mon_AM_NICHOLAS H GOH MAOWEN_475_Archive
L3_Mon_AM_NIKHIL SHASHIDHAR_992_Archive
L3_Wed_PM_NING ZHI YAN_096_Archive
L3_Wed_AM_OH YI XIU WILSON_510_Archive
L3_Mon_AM_ONG CHUAN KAI_208_Archive
L3_Mon_AM_ONG HEE JET_579_Archive
L3_Wed_AM_ONG JUN LIN JEREMIAH_679_Archive
L3_Mon_AM_ONG SHAO YONG_220_Archive
L3_Wed_PM_OONG JIN RONG JARED_178_Archive
L3_Wed_AM_OU NINGXIANG_191_Archive
L3_Mon_AM_OW YONG JIN XUAN_649_Archive
L3_Wed_PM_PARANJAPE INDRANEEL RAJEE_412_Archive
L3_Wed_PM_POOBALAN AATMIKA LAKSHMI_701_Archive
L3_Mon_AM_RACHEL FONG RUI YUAN_567_Archive
L3_Wed_PM_REYNOLD SAMEL LAM_216_Archive
L3_Mon_AM_RICHARD KURNIAWAN_030_Archive
L3_Wed_PM_RICHARD LOONG CHENG JUN_496_Archive
L3_Wed_PM_RYAN TAN_565_Archive
L3_Wed_PM_RYUJI KOW JIE SI_117_Archive
L3_Wed_AM_SAMUEL TAN SZE WEE_575_Archive
L3_Wed_AM_SAUNG NAYCHI MIN_779_Archive
L3_Wed_PM_SEBASTIAN SOEWANTO_268_Archive
L3_Mon_AM_SEET SZE WEN_125_Archive
L3_Mon_AM_SENTHILKUMAR SRIRAM_934_Archive
L3_Mon_AM_SEOW RUI SHENG_454_Archive
L3_Wed_PM_SHAN YUXUAN_473_Archive
L3_Wed_AM_SHANNEN TAN_713_Archive
L3_Mon_AM_SIM JUSTIN_898_Archive
L3_Wed_AM_SIM QIAN HUI_289_Archive
L3_Mon_AM_SIUT WAI HONG CLEMENT_175_Archive
L3_Wed_PM_SONG ZIJIN_699_Archive
L3_Mon_AM_STEVEN ANTYA ORVALA WASKI_459_Archive
L3_Wed_PM_SURESH ABIJITH RAM_215_Archive
L3_Wed_AM_SYED OMAR ZORAN_260_Archive
L3_Wed_AM_TAN HSIEN RONG_011_Archive
L3_Wed_AM_TAN JIN SHENG BRIAN_260_Archive
L3_Mon_AM_TAN JUN HAO KENNETH_519_Archive
L3_Wed_AM_TAN TZE LOONG_867_Archive
L3_Mon_AM_TAN WAN LIN_587_Archive
L3_Wed_AM_TAN YI ZHE_674_Archive
L3_Mon_AM_TAN YU XIANG GARETH_682_Archive
L3_Wed_AM_TAN ZI XI_046_Archive
L3_Wed_AM_TANG JUN MEI SHANICE_739_Archive
L3_Wed_PM_TAY JIUN YUAN_398_Archive
L3_Mon_AM_TAY YU YANG_039_Archive
L3_Wed_AM_TENG YUANKAI_862_Archive
L3_Mon_AM_TEO JUN YONG ADEN_600_Archive

L3_Mon_AM_TEOH JING YANG_566_Archive
L3_Wed_AM_THANT AUNG HTET NYAN_654_Archive
L3_Wed_AM_TIAN SHIXING_830_Archive
L3_Mon_AM_TOH HONG FENG_218_Archive
L3_Mon_AM_TOH MING CHUN_573_Archive
L3_Mon_AM_TONG ZHENG HONG_407_Archive
L3_Mon_AM_TRICIA BOO KOH WEI PING_477_Archive
L3_Mon_AM_TU HUIYU_423_Archive
L3_Wed_PM_UDAYAKUMAR NIVETHA_182_Archive
L3_Wed_PM_VARATHARAJU VIGNESH_677_Archive
L3_Wed_AM_VARNIKA SRIVASTAVA_513_Archive
L3_Wed_PM_VISHNU_211_Archive
L3_Wed_AM_VU VAN DUNG_825_Archive
L3_Wed_AM_WANG HAOYANG_534_Archive
L3_Wed_AM_WANG SILANG_130_Archive
L3_Mon_AM_WANG TINGJIA_501_Archive
L3_Wed_AM_WANG YONGBIN_968_Archive
L3_Wed_PM_WILSON LEE JUN WEI_735_Archive
L3_Wed_AM_WINSTON LIM CHER HONG_359_Archive
L3_Wed_AM_WONG ZHONG XIANG_929_Archive
L3_Mon_AM_WOO KAI NING_128_Archive
L3_Wed_PM_WOO WEN JUN_636_Archive
L3_Wed_PM_WU ZHEN_039_Archive
L3_Wed_PM_YAO HE_485_Archive
L3_Mon_AM_YEO MENG HAN_360_Archive
L3_Mon_AM_YONG SHAN LING_973_Archive
L3_Wed_PM_ZENG ZIQIU_720_Archive
L3_Wed_PM_ZHAI YUXIN_744_Archive
L3_Wed_AM_ZHANG WENZE_047_Archive
L3_Wed_AM_ZHANG ZHITONG_202_Archive
L3_Wed_AM_ZHAO LIXIUQI_464_Archive
L3_Wed_AM_ZHOU HUIQI_364_Archive
L3_Mon_AM_ZHOU KAIWEN_651_Archive