DATAPATH WITH CONTROL

This lab has you create a datapath with FSM control.

BACKGROUND

Review the notes on creating datapath+control designs.

TASK DESCRIPTION

In the previous two labs, you implemented the following fixed-point computation (K1, K2, K3 are constants):

$$Y = (X1 * K1) + (X2 * K2) + (X3 * K3)$$

You are to implement the exact same computation, except you now have the following limitations:

- One input bus
- One multiplier (you must use a hardmacro multiplier)
- One adder
- The critical path of the design cannot include both the multiplier and the adder (the multiplier output cannot be connected to the adder input)
- The computation must be implemented in four clock cycles, with the X1, X2, X3 values arriving in the first three cycles.
- The inputs are unregistered, the outputs are registered as per our datapath conventions.

Since this is the first design with a FSM and registers, the complete implementation is provided for you, all you to have to do is translate it to Verilog.

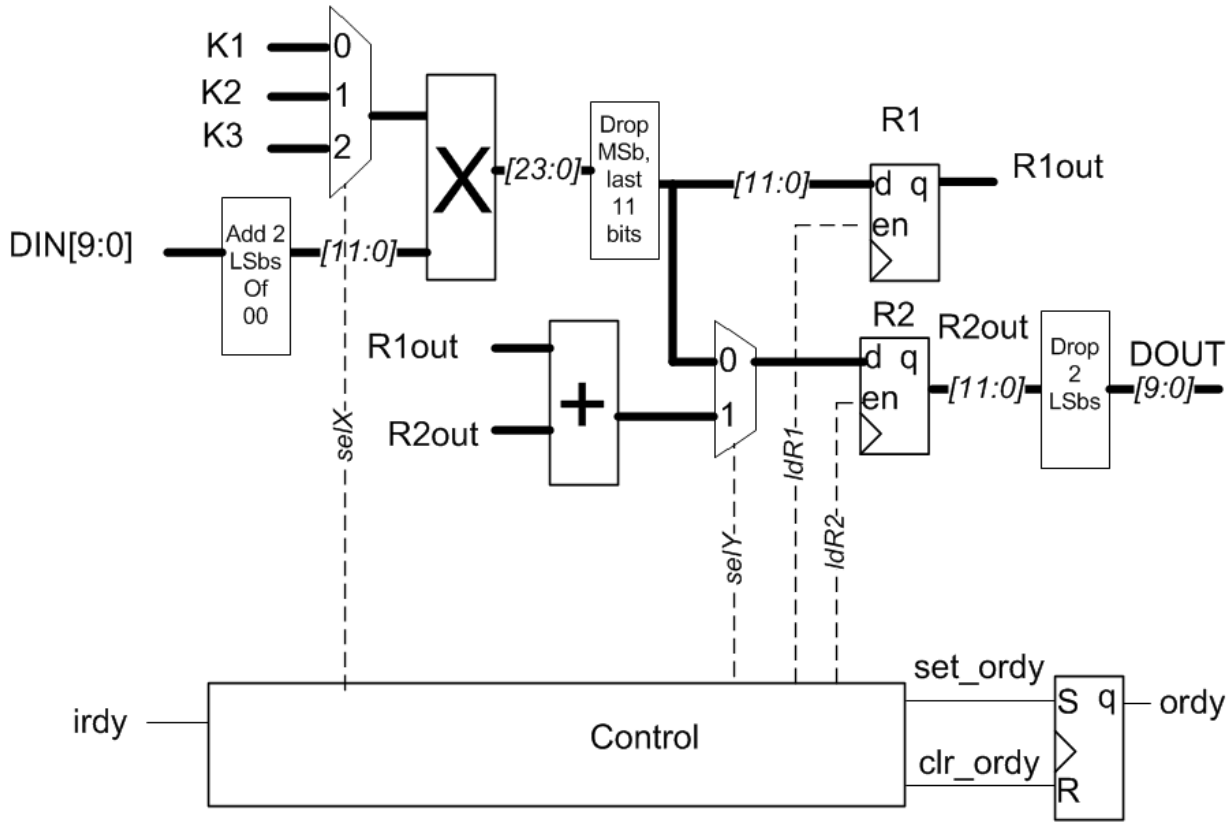
Scheduling Table

Clock Cycle	IRDY	DIN	Mult	Adder	R1	R2	Ordy	Dout
1	1	X1	X1*K1		Load from Multiplier		0	
2	0	X2	X2*K2			Load from Multiplier	0	
3	0	X3	X3*K3	R1+R2	Load from Multiplier	Load from Adder	0	
4	0			R1+R2		Load from Adder	0	

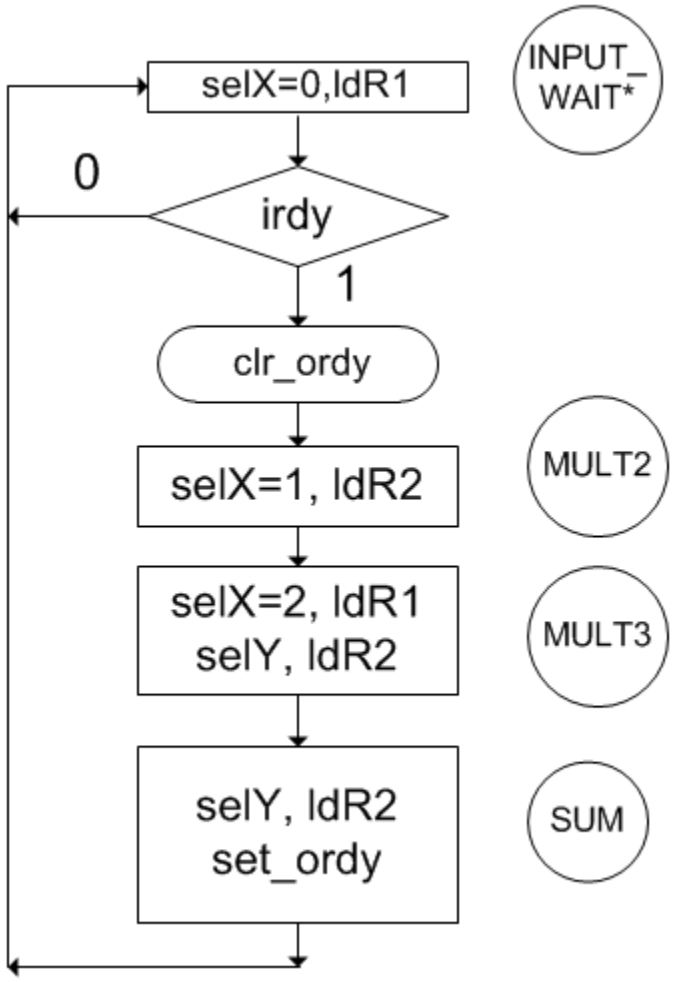
5	1	X1	X1*K1		Load from Multiplier		1	First Result
6	0	X2	X2*K2			Load from Multiplier	0	
7	0	X3	X3*K3	R1+R2	Load from Multiplier	Load from Adder	0	
8	0			R1+R2		Load from Adder	0	
							1	Second Result

Latency of Design is 5 clocks (from first input to output ready)
Throughput of Design is 4 clocks (new inputs accepted every four clocks).

Datapath:



ASM Chart:



The module interface is shown below. The definition of the *irdy*, *ordy* handshaking signals are the same as used in the control+datapath notes (*irdy* is asserted for one clock cycle when the first data input is available, *ordy* is asserted when the result is ready and stays asserted until the next *irdy* assertion). The output databus contains the output value until the next *irdy* assertion.

```
module lab6dpath(reset, clk, irdy, ordy, din, dout  );
 input reset, clk, irdy;
 input [9:0] din;
 output [9:0] dout;
 output ordy;
```

Design Deliverable (Part 1)

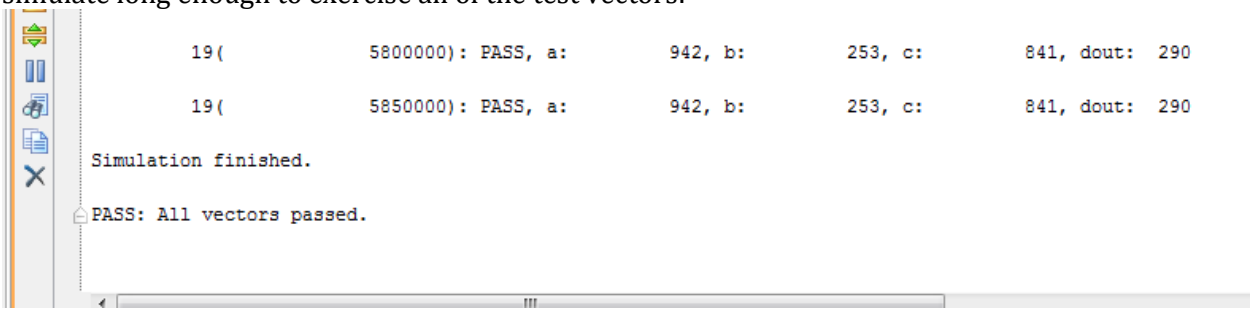
The ZIP archive named lab6_files_part1 associated with this lab contains the following files:

- *lab6dpath.v* -- complete this module
- *tb_lab6dpath.v* – test bench
- *multadd_vectors.txt* – input vector file for testbench.

- *report.docx* – report file that needs to be filled out

Implement your Design Deliverable in a directory named *lab6_part1* and a project named *lab6_part1* and test it with the supplied testbench. Your design must be synthesizeable and contain no latches after synthesis. Your design is expected to simulate correctly in both behavioral simulation and post-route simulation.

When you run the simulation testbench, you must execute the command 'run 5 us' in order to simulate long enough to exercise all of the test vectors:



There are a total of 19 test vectors:

Fill out the requested information for the *Implementation* deliverable in the *report.doc* file.

Design Deliverable (Part 2)

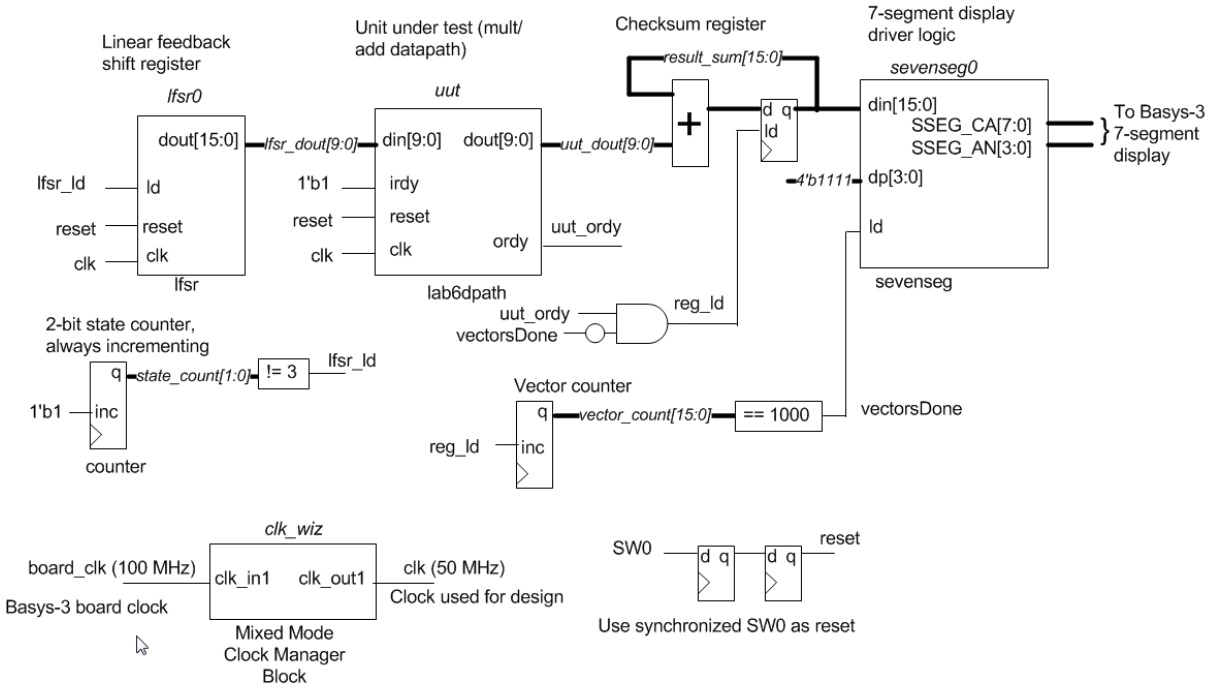
This deliverable requires use of your Basys 3 board.

The ZIP archive named lab6_files_part2 associated with this lab contains the following files:

- Verilog design files: *hw_testbench.v*, *lfsr.v*, *sevenseg.v*, *pulsegenms.v*
- Verilog simulation testbench file: *tb_hw_testbench.v*
- Constraint file: Basys3_Master.xdc

The above design files implement hardware tester of the datapath for Part1. The *hw_testbench.v* design file is the top level module. It applies 1000 pseudo-random vectors to the datapath using a Linear Feedback Shift Register (LFSR), and accumulates a checksum of the result that is displayed on the 7-segment display on the Basys-3 board. The correct hex value for the checksum for the 1000 vectors is 0xbb26. SW0 on the board is used as a reset signal for the design. The testbench file (*tb_hw_testbench.v*) is be used to simulate the design before downloading into the Basys-3 board.

The schematic of the hardware tester is shown below:

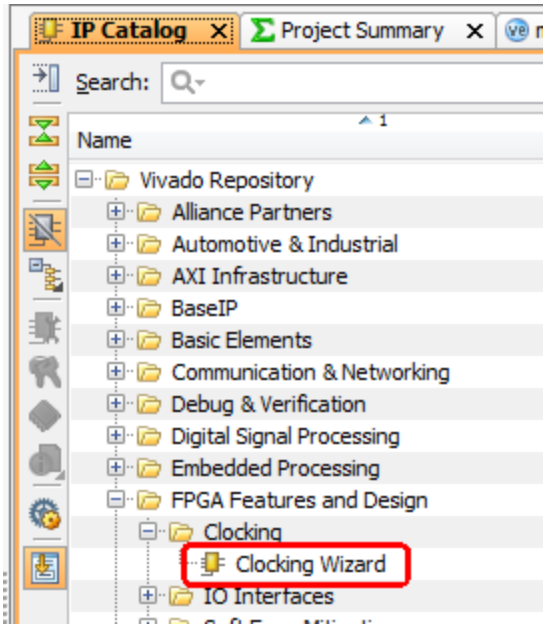


Comments on the hardware tester:

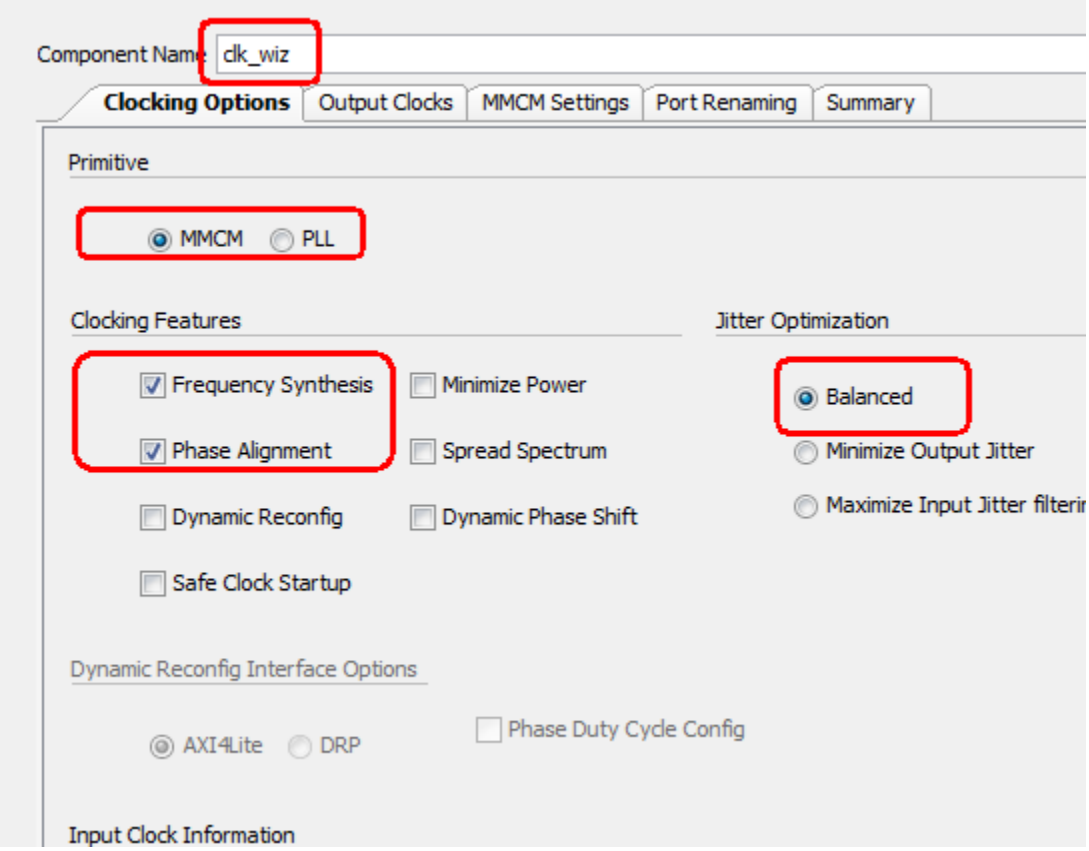
- The reset is provided by SW0 of the Basys-3 board after being synchronized through 2-DFFs.
- A mixed-mode clock manager block is used to generate a 50 MHz system clock from the 100 MHz clock provided on the Basys3-board.
- Pseudo-random input vectors are generated by the lfsr module (linear feedback shift register). A new LFSR value is generated each time the *ld* input is high. The control logic asserts this input for the first three clock cycles of the four clocks that the mult/add datapath is active by using a 2-bit state counter. The LFSR module produces a maximum length sequence of 65535 values before repeating (all values are produced in the range 0 to 65535 except 0).
- The input rdy signal to the unit under test (UUT, the mult/add datapath) is always high so the datapath is in constant operation. Operation starts immediately after reset.
- A 16-bit checksum accumulator (register+ adder) is used to accumulate the 10-bit output of the UUT; this is loaded each time the UUT ordy signal is asserted and 1000 vectors have not been reached (after 1000 vectors, the checksum register stops loading). A 16-bit counter is used to keep track of the number of vectors generated, this is incremented with the same signal used to load the checksum register.
- The16-bit output of the checksum register is the input to the 7-display logic that displays this 16-bit output as 4-digit hex number. The 8-bit cathode (SSEG_CA) and 4-bit anode (SSEG_AN) are connected to the 7-segment display driver. The 7-segment display is logic is loaded with the output of the checksum register once 1000 vectors have been processed.

Procedure

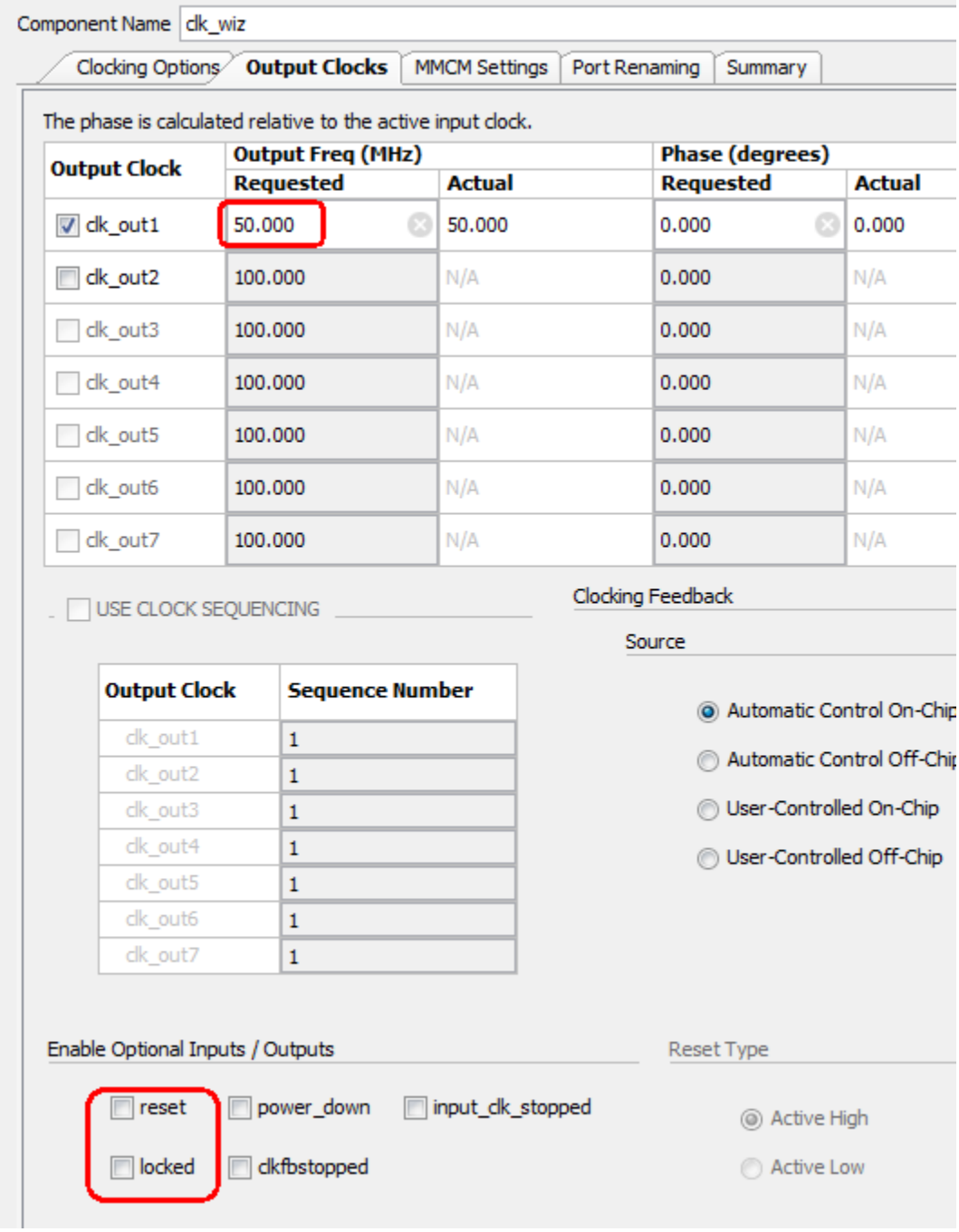
1. Create a new directory named *lab6_part2*, and a project named *lab6_part2* and add the above files to it. Also copy your Verilog solution file into this directory and re-instantiate your multiplier using IP Catalog.
2. Use the Clocking Wizard from the IP Catalog to generate a Mixed Mode Clock Manager (MMCM) block named 'clk_wiz' that will generate a 50 MHz clock from the Basys-3 100 MHz clock following the steps below.



Edit the component name to be 'clk_wiz' and set the other options as shown.



In the 'Output Clocks' tab, edit the requested clock frequency for clk_out1 to be 50 MHz, and uncheck the 'reset' and 'locked' optional inputs/outputs.



Generate this block as it is used in the *hw_testbench.v* file.

3. Run the behavioral simulation. Add the signals from the uut, restart the simulation, and run for 10 us. On the waveforms, find where the *uut/vector_count* signal is '50' (decimal value) and verify that the *uut/result_sum* value is 0x612e (hex value) – see the following screenshot. This means that after 50 randomly generated vectors, the result checksum is 0x612e. If you do not get this, then something is wrong -- re-check the steps up to this point and verify that you followed the steps faithfully. If you cannot determine the problem, then

seek assistance. Capture a simulation screen shot that shows the checksum after 100 vectors have been applied and include this in your report.

4. From within Vivado, do 'Run Implementation', then 'Generate Bitstream'. Use the Hardware Manager to download this design into your board (review the steps in Lab1 for downloading to your board). The digits 'bb26' should appear on your 7-segment LED – this is the checksum after 1000 vectors have been applied to the design. Get the TA to check off your board.

For submission, create a directory named 'lab6_*netid*', i.e., (lab6_rbr5). Copy both the lab6_part1 and lab6_part2 directories to this directory.

Copy your completed report.docx to this directory.

Create a ZIP archive of this directory and submit it to Canvas.