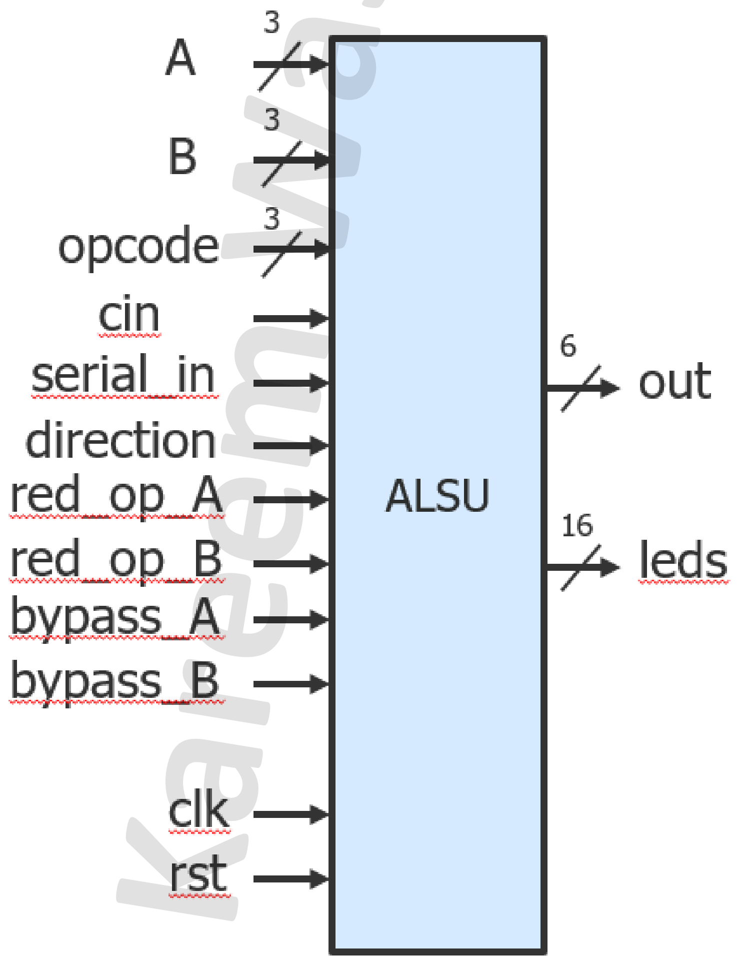
Assignment 4

Design the following circuits using Verilog and create a testbench for each design to check its functionality. Create a do file for the questions. **Check the deliverables of each question at the end of the document.**

1) ALSU is a logic unit that can perform logical, arithmetic, and shift operations on input ports

- Input ports A and B have various operations that can take place depending on the value of the opcode.
- Each input bit except for the clk and rst will be sampled at the rising edge before any processing so a D-FF is expected for each input bit at the design entry.
- The output of the ALSU is registered and is available at the rising edge of the clock.



Inputs

Each input bit except for the clk and rst will have a DFF in front of its port. Any processing will take place from the DFF output.

Input	Width	Description
clk	1	Input clock
rst	1	Active high asynchronous reset
A	3	Input port A
B	3	Input port B
cin	1	Carry in bit, only valid to be used if the parameter FULL_ADDER is "ON"
serial_in	1	Serial in bit, used in shift operations only
red_op_A	1	When set to high, this indicates that reduction operation would be executed on A rather than bitwise operations on A and B when the opcode indicates AND and XOR operations
red_op_B	1	When set to high, this indicates that reduction operation would be executed on B rather than bitwise operations on A and B when the opcode indicates AND and XOR operations
opcode	3	Opcode has a separate table to describe the different operations executed
bypass_A	1	When set to high, this indicates that port A will be registered to the output ignoring the opcode operation
bypass_B	1	When set to high, this indicates that port B will be registered to the output ignoring the opcode operation
direction	1	The direction of the shift or rotation operation is left when this input is set to high; otherwise, it is right.

Outputs and parameters

Output	Width	Description
leds	16	When an invalid operation occurs, all bits blink (bits turn on and then off with each clock cycle). Blinking serves as a warning; otherwise, if a valid operation occurs, it is set to low.
out	6	Output of the ALSU

Parameter	Default value	Description
INPUT_PRIORITY	A	Priority is given to the port set by this parameter whenever there is a conflict. Conflicts can occur in two scenarios, red_op_A and red_op_B are both set to high or bypass_A and bypass_B are both set to high. Legal values for this parameter are A and B
FULL_ADDER	ON	When this parameter has value "ON" then cin input must be considered in the addition operation between A and B. Legal values for this parameter are ON and OFF

Opcodes & Handling invalid cases

Opcode	Operation
000	AND
001	XOR
010	Addition
011	Multiplication
100	Shift output by 1 bit
101	Rotate output by 1 bit
110	Invalid opcode
111	Invalid opcode

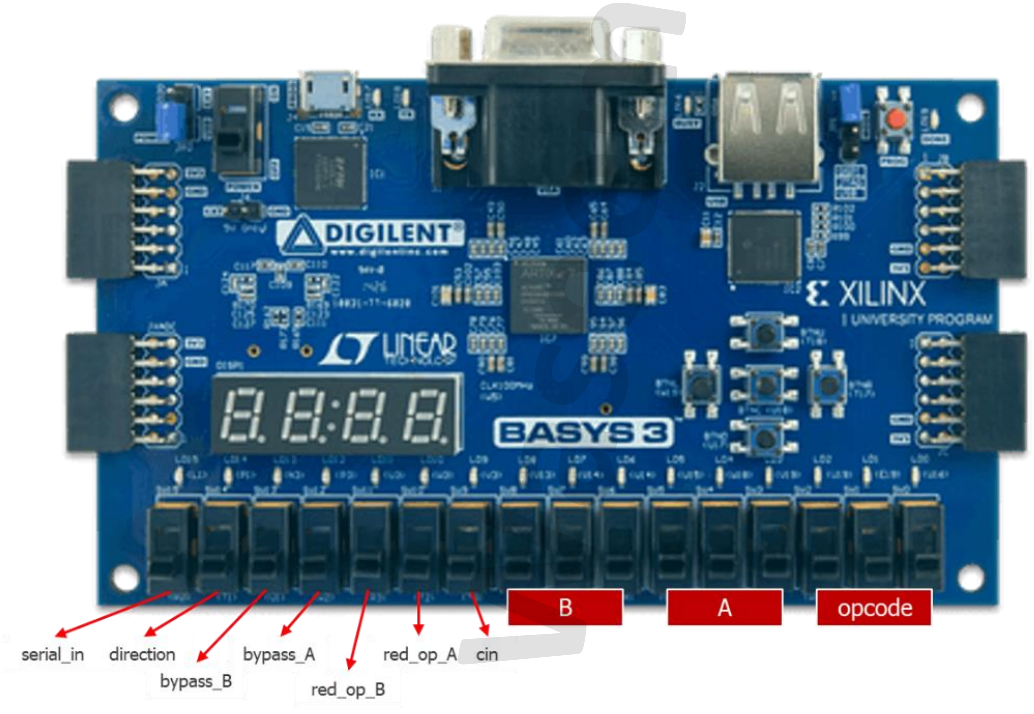
Invalid cases

1. Opcode bits are set to 110 or 111
2. red_op_A or red_op_B are set to high and the opcode is not AND or XOR operation

Output when invalid cases occurs

1. leds are blinking
2. out bits are set to low, but if the bypass_A or bypass_B are high then the output will take the value of A or B depending on the parameter INPUT_PRIORITY

You are required to write the constraint file for the ALSU. Refer to the below figure to know the physical constraints.



- "clk" is connected to W5 pin as suggested in the board's reference manual with frequency 100 MHz
- "rst" is connected to button U18
- "leds" are connected to the LEDs on the board

You are required to set a debug core to be able to debug any input or output for the design.

<u>**Note:**</u>

- You can use in the testbench the system function "$urandom_range" which returns randomized unsigned integer within a range if you want to strict the opcode to have valid opcodes only
    - signal_in = $urandom_range(5,15); //randomized between 5 and 15
- Bit stream generation will not be successful since the output "out" is not connected yet (You will connect them to the seven segment display in assignment 5)

<u>**Testbench for this design will be as follows:**</u>

1. Parameter will not be overridden so we will keep the design with its default values

2. Stimulus Generation (Initial Block) – Driving only valid opcodes

2.1 Verify Asynchronous rst Functionality

- Drive rst to **1**.

- Force remaining inputs to zero
- Wait for the **negative edge of the clock**.
- Add a condition for **output out and leds** to enable self-checking.

2.2 Verify Bypass Functionality

- Drive bypass_A and bypass_B to 1 and deactivate the rst
- Randomize A, B, and opcode (opcode valid values only using $urandom_range) inside a for loop.
- Wait for **two negative edges of the clock** (since the design has 2 stages of registers)
- Self-check **output out** for correctness.

2.3 Verify opcode 0 Functionality

- Drive bypass_A, bypass_B, and opcode to **0**.
- Randomize A, B, red_op_A and red_op_B inside a for loop.
- Wait for two **negative edges of the clock**.
- Self-check **output out** for correctness.

2.4 Verify opcode 1 Functionality

- Drive opcode to 1.
- Randomize A, B, red_op_A and red_op_B inside a for loop.
- Wait for two **negative edges of the clock**.
- Self-check **output out** for correctness.

2.5 Verify opcode 2 Functionality

- Drive opcode to 2 and drive red_op_A and red_op_B to 0.
- Randomize A, B, and cin inside a for loop.
- Wait for two **negative edges of the clock**.
- Self-check **output out** for correctness.

2.6 Verify opcode 3 Functionality

- Drive opcode to 3.
- Randomize A, B inside a for loop.
- Wait for two **negative edges of the clock**.
- Self-check **output out** for correctness.

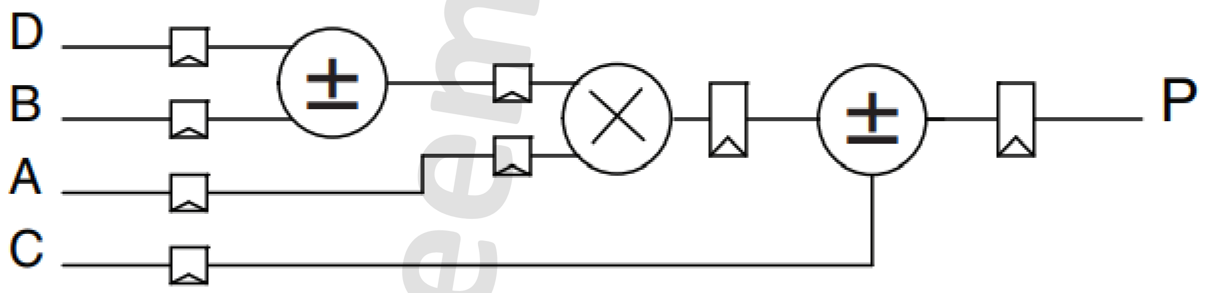
2.7 Verify opcode 4 Functionality

- Drive opcode to 4.
- Randomize A, B, direction and serial_in inside a for loop.
- Wait for two **negative edges of the clock**.
- Check the waveform for **shifting functionality**

2.8 Verify opcode 5 Functionality

- Drive opcode to 5.
- Randomize A, B, direction and serial_in inside a for loop.
- Wait for two **negative edges of the clock**.
- Check the waveform for **rotating functionality**

2) Design the following simplified version of the DSP block DSP48A1 from Xilinx Spartan-6 FPGA. Use a directed testbench to simplify the verification. Check the waveform to make sure that the operations are done correctly at every pipeline stage. Run vivado and add a constraint file that has **only timing constraint** which defines the clock frequency to 100 MHz to pin W5 as usual. Since the design has I/O ports more than the Basys 3 FPGA board, choose the FPGA part xc7a200tffg1156-3 to accommodate for the large I/O ports.



Port	Type	Width
A	Input	18
B	Input	18
C	Input	48
D	Input	18
clk	Input	1
rst_n (sync active low)	Input	1
P	Output	48

Parameters:

- OPERATION: take 2 values either “ADD” or “SUBTRACT”, Default value “ADD”

- When subtracting use “D - B” and “multiplier_out – C”. multiplier_out is an internal signal

3) Design a Time Division Multiplexing (TDM) block that takes four 2-bit inputs and outputs each one sequentially on every clock cycle.

Time Division Multiplexing (TDM) is a technique used in digital communication and processing systems to efficiently share a single communication channel or port among multiple data paths. A TDM Multiplexer (MUX) achieves this by transmitting each input data path to the output path in a fixed, repeating sequence. This allows multiple signals to share the same output line.

1. The module should have the following inputs and outputs:
   - Inputs:
     - clk
     - rst: Active-high asynchronous reset signal
     - in0, in1, in2, in3: Four 2-bit input channels
   - Output:
     - out: 2-bit output that sequentially transmits each input on every clock cycle
2. Internal counter will cycle over 0, 1, 2 and 3, transmitting in0, in1, in2, and in3 respectively. Incase of reset, counter output is zero and so in0 will be connected to the output.
3. Provide a testbench to verify the correct operation of your module. The testbench should include:
   - Clock and reset generation
   - Self-checking test cases covering normal operation and reset behavior
4. Run Vivado and define the following constraints
   - Clk set to 100 MHz
   - Rst assigned to a button on the FPGA
   - In0, in1, in2, in3 assigned to switches on the FPGA
   - Output out assigned to LEDs on the FPGA

<u>Deliverables:</u>

1) The assignment should be submitted as a PDF file with this format <your_name>_Assignment4 for example Kareem_Waseem_Assignment4.
2) RTL and Testbench code snippets
3) Snippets from the waveforms captured from QuestaSim for the design with inputs assigned values and output values visible.
4) Snippets from the RTL schematic after the elaboration
5) Snippets from the technology schematic after synthesis
6) Snippet from the utilization & timing report after the synthesis and implementation.
7) Snippet of the “Messages” tab showing no critical warnings or errors after running elaboration, synthesis, and implementation.
8) [Optional] Linting snippet from the linting tool showing no warnings or errors (use the default methodology and goals)

Note that your document should be organized as follows:

1. Verilog Code
    a. RTL Code
    b. Testbench code)
2. Simulation Tool
    a. Do file
    b. QuestaSim Snippets)
3. Synthesis Tool
    a. Constraint File showing the debug core added to the end of the file (Question 1 only)
    b. Elaboration (“Messages” tab & Schematic snippets)
    c. Synthesis (“Messages” tab, Utilization report, timing report & Schematic snippets)
    d. Implementation (“Messages” tab, Utilization report, timing report & device snippets)
4. Linting Tool [Optional]
    a. Run lint on the RTL and take a snippet showing no warnings or errors