Verification plan & Subroutines

Requirements for all testbenches:

- Create a verification plan -> Use this verification plan as a template
- Create a self-checking testbench either by calculating the correct result as done in the class or creating a reference in the testbench that generates the expected result to check the functionality.
- Task for checking the normal operation of the design.
- Task to reset and check the reset functionality.
- Counters to keep track of the correct and error count. Display them by the end of your testbench.

Deliverables:

One PDF file having screenshots of the following:

1. Design
    a. If the design has bugs, then fix them and mention what you have fixed
2. Verification Plan
3. Testbench
4. Do file
5. Coverage report text file
    a. Add this option (-du <design_module_name>) to the coverage save command to save only the coverage of the design and exclude the testbench
1. **Clear and neat** QuestaSim waveform snippets showing the functionality of the design with **each verification plan label (column A) mentioned in the verification plan**
2. Branch, statement and toggle coverage report snippets with justification if you could not reach 100% coverage for the designs and add code coverage exclusions to reach 100% coverage (watch the demo shared on Google classroom to learn more about coverage exclusions)

Questions:

1) Add more test vectors to the adder testbench done in the class to reach 100% code coverage.

2) Verify the functionality of the following 4 to 2 priority encoder The design consists of 3 inputs which are the following:

- 4-bit input D
- clk
- synchronous active high rst which resets the outputs to 0

The design has 2 output ports (2-bit output Y and 1-bit output valid) which update with the positive edge of clock. Output valid is set to 1 when more than one input line is high. If all the inputs are '0', then valid output is zero. In this case, the output Y is considered as don't care conditions denoted by 'X'.

D3	D2	D1	D0	Y1	Y0	valid
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Since the design is simple, it is required to perform exhaustive verification which means generating all possible input combinations for the input D and make sure that the outputs are correct.

3) ALU design will be provided and has the following characteristics.

- Reset which resets C to 0.
- 4-bit signed inputs, A and B
- 5-bit registered signed output C
- 4 op-codes
    - add
    - sub (A-B)
    - bitwise invert input A
    - reduction OR input B

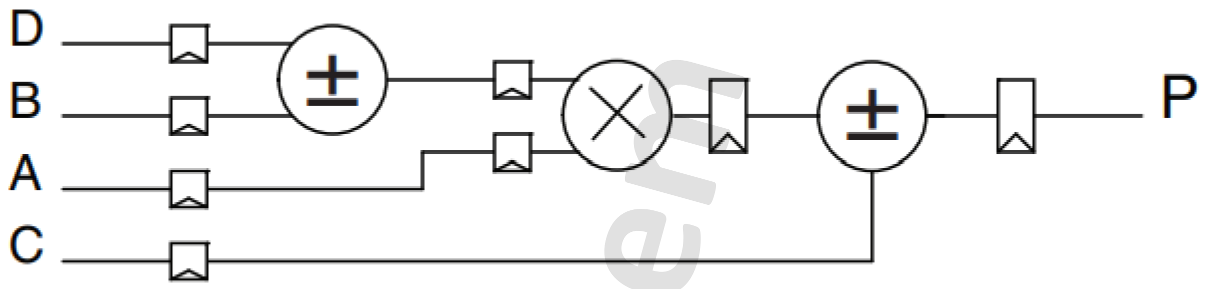
Assume the following encoding of the opcodes.

Opcode	**Encoding**
Add	2'b00
Sub	2'b01
bitwise invert input A	2'b10
reduction OR input B	2'b11

Do a directed verification the same we did in the class with the adder to apply test vectors hitting the boundaries (extreme values) for the addition and subtraction and think how to approach the bitwise invert and reduction OR to make sure the output is correct.

4) Verify the functionality of the following simplified version of the DSP block DSP48A1. All the registers are positive-edge triggered with the clock and have an async active low reset. Assume that we will only use the DSP in the addition mode only. For simplicity, you can drive the DSP inputs every 4 clock cycles to check the P output. Use a for loop and randomize the inputs after the reset checking.

Note: it is an unsigned DSP block.



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

Parameters:

1. 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