Prac 3 – Arithmetic Circuits in VHDL

NOTE: Things can take time. So be prepared with the lab activity, do all the preliminary designs and as much implementation as you can beforehand.

Design Task

Part A – 16-bit two's complement adder with status flags and saturation

Inputs: A (16 bits), B (16 bits), C_IN (1 bit), SAT_EN (1 bit)
Outputs: SUM (16 bits), C (1 bit), N (1 bit), Z (1 bit), V (1 bit)

The first design task for this practical is to develop a 16-bit adder that takes a carry in and 2 signed two's complement 16-bit numbers A and B. This adder also has an enable pin for saturation functionality (SAT_EN). When this SAT_EN pin is disabled ('0'), the adder will return the sum of the two inputs plus the carry in.
When the SAT_EN pin is enabled ('1'), the adder will return the sum of the two inputs plus the carry in unless there is an overflow, in which case the output will be saturated to either positive or negative extreme of the 16-bit two's complement range. That is, a negative overflow will saturate to 0x8000 and a positive overflow will saturate to 0x7FFF.
The adder will also have the following output status flags:

N for negative output,
Z for zero,
C for carry out and
V for overflow.

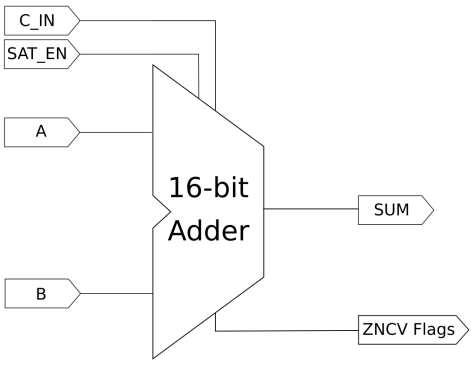


Figure 1: 16-bit adder with status flags and saturation

1. Design the above 16-bit adder with saturation using VHDL. You can use any abstraction level of your choice, however the design should not be overly complicated.
2. Provide sufficient simulation output to show that the design is functionally correct. i.e., it produces the correct sum and correct status flags and also it is able to saturate the answer if saturation is enabled and overflow occurs. Saturation on both positive and negative sides should be demonstrated in the simulation.
3. Provide the RTL and synthesis schematics as well as the FPGA resource consumption from the synthesis output

Part B -Binary Coded Decimal Adder

Develop a Binary Coded Decimal (BCD) Adder to add two BCD digits. The BCD adder will add BCD digits and output the resulting sum in BCD. The algorithm for a one-digit BCD adder is given below.

The algorithm for a BCD adder requires the adder to carry out and overflow once the inputs sum up to hexadecimal ”A” (10 in base 10). Using hexadecimal values, this is equivalent to adding 6 to any value over 9, forcing an overflow and a carry out.

Hence, the pseudocode for this is as follows.

```
If A + B > 9:
        Carry = 1
        Output = A + B + 6
Else
        Carry = 0
        Output = A + B
```

Why does the BCD Adder algorithm use a factor 6? Think how this will work with A and B being 4-bit unsigned numbers. Would the algorithm still work for the case "(8 + 8)" or "(9 + 8)"? What must be added to allow this module to be cascaded?

A block design of one-digit BCD adder is presented in Figure 1. You will need to design the correction logic.

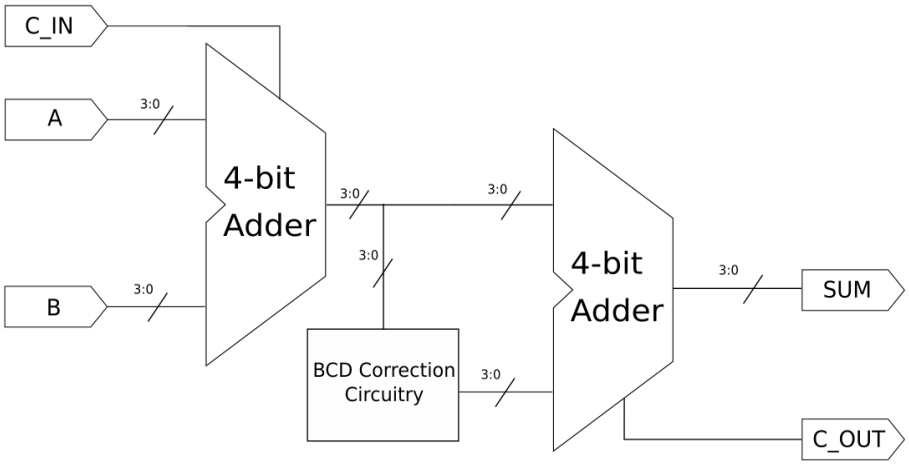
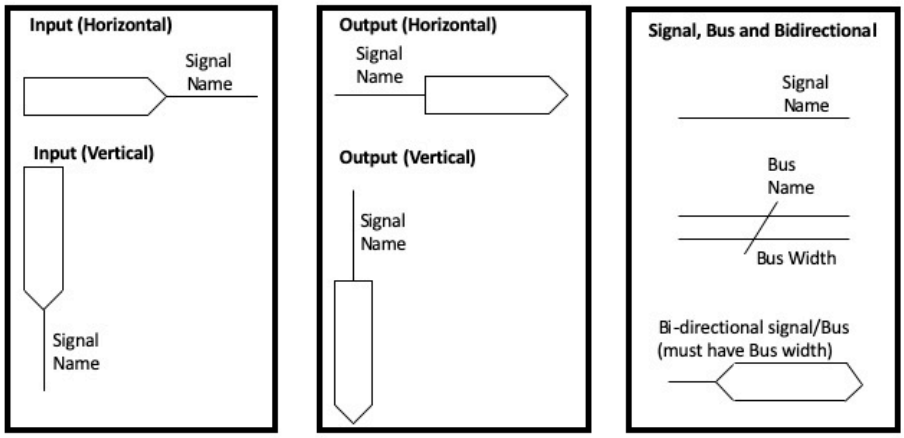


Figure 2: One Digit BCD Adder Block Diagram.

1. Create a 1-digit BCD adder module in VHDL using a behavioural (with process statement) description and make sure that the module is functioning correctly (simulate and verify).
2. Use the above 1-digit BCD module as a building block to design a 2-digit BCD adder using a structural approach to connect the building blocks together. Note that you need 3 BCD digits to output the result.
3. You should be able to detect non-BCD inputs and output a 0 if detected.
4. Fully simulate your 2-digit BCD adder to show that it is functioning correctly.
5. Analyse the synthesis output (synthesis schematic) by comparing with a manual design and identify how the synthesis tool mapped your description into hardware.

Report Content and Format

A typeset PDF report must be submitted by the due date with the following content:
Introduction stating the problem description, any assumptions, and design objectives
Design description including a block diagram explaining the complete design. Block diagrams should be drawn using standard symbols for inputs/outputs, signals and busses as shown below.



- Detailed simulation results showing the key scenarios to prove that the design is functionally correct and delivers the expected output.
- Register transfer level (RTL) schematic of the design obtained from Vivado tools (you should try to back-annotate the RTL schematic and try to identify the basic sub-systems/blocks in your design)
- Synthesis schematic and results including FPGA resource consumption
- Conclusion – a reflection of what you've achieved in this exercise, problems (if you had any) and potential improvements to your design (if any)
- References (if any)
- You must have the marking sheet on the next page as the last page of your report and your marks will be indicated on this sheet (this has also been uploaded on BB as a single pdf file so that you can append this to your report as the last page)
- If you don't complete all the tasks by the due date, you can still submit the report explaining what you have achieved, and this will be treated as an attempt to the practical. Note that you need to attempt all the practicals to pass the course.

Marking Criteria

The pracs in this course are marked to a specific criteria. This means you must demonstrate sufficient understanding and functionality and the marking will be done according to the rubric provided below. You must attempt each prac (i.e., a report must be received by the due date) to pass the course. All designs VHDL code used **must be your own work.** You are NOT permitted to use other VHDL code sources, unless directed to. Plagiarism is unacceptable and please read and understand the School Statement on Misconduct, available on the ITEE website at: http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism . To avoid problems make sure that your VHDL code is the product of your work and do not let anybody else 'reuse' your code.

Marks	**Criteria**
Simulation	
0	Simulation not attempted or does not work for both part A and part B
1	Simulation partially works for either part A or part B
2	Simulation partially works for both part A and part B
3	Simulation fully works for either part A or part B
4	Simulation fully works for both part A and part B
Report	
0	No evidence of content or work
1	Some content, insufficient explanation of circuit
2	Reasonable content, some explanation of circuit
3	Good content, reasonable explanation of circuit
4	Excellent content, good explanation of circuit
Oral assessment	
0	No knowledge of the design
1	Very little knowledge of the design
2	Reasonable knowledge of the design
3	Good knowledge of the design.
4	Excellent knowledge of the design.
Total (12):	**Marker Initials:** **Date:**