EG3204 FPGA Assignment 1
A Programmable 4-bit Counter

1 Introduction

In the next two weeks you will design, implement, and simulate a programmable up/down 4-bit counter. The counter will keep track of a value, will be able to enter either count up or count down mode, and will have an programmable upper limit (modulo) that it can count to before cycling back to zero. When the counter is enabled (active), it will increment or decrement the counter once per second. In other words you need to design a modulus N (MOD-N) counter where N will be set using a 4-bit input bus.

2 Individual exercise

This is an assessed exercise. Your solution will constitute 25% of your overall mark for EG3204. You may work on your solution in the lab hours in weeks 7 and 8, and you may also work on it in your own time.

While you are encouraged to discuss aspects of the design with others, you must attempt the writing of the VHDL as an individual exercise. You may not share your VHDL solution, or part of your solution, with other students or report. Any submissions (or parts of solutions) that are found to be shared between students will be dealt with under the University policy for collusion and/or plagiarism, and may result in a module mark of zero and/or loss of credit for the module or even downgrading of your final degree. Details of what you are required to submit, how you should submit your work, and deadlines for submitting your work are given at the end of this sheet.

During the laboratory sessions for weeks 7 and 8 reduced demonstrator support will be available but because this is an individual exercise demonstrators will not be debugging code.

3 Design of the counter

Our Counter is a sequential circuit, where the value of the counter is incremented or decremented once every second whenever an Enable button is pressed on the development board. The counter does not change when the enable button is not pressed. When the Direction input is zero (false) the counter will count up, and when it is one (true) the counter will count down. The counter can be reset (set back to zero) when the Reset input is one. Additionally, the counter can have a maximum (modulo) value programmed using a 4-bit switch configuration. This value is also set when the Reset button is pressed. The counter has six input signals and a single output signal:

1. **clock** : A standard 1-bit 50 MHz clock signal.
2. **clk_div** : A second clock signal that pulses once every second (1 Hz).
3. **reset** : A standard 1-bit reset line, that puts the counter back into an initial state. You should decide whether a synchronous or asynchronous reset is appropriate.
4. **enable_count** : The signal that causes the counter to increment or decrement when high.
5. **direction** : A 1-bit signal that causes the counter to count up or down depending on its value.
6. **mod_value**: A 4-bit bus (STD_LOGIC_VECTOR) that is used to set the maximum value the counter can count up to, in the range 0-15. The value on this bus should be set when the counter is reset.
7. **output_counter** : A 4-bit bus (STD_LOGIC_VECTOR) that outputs the current stored value in the counter.

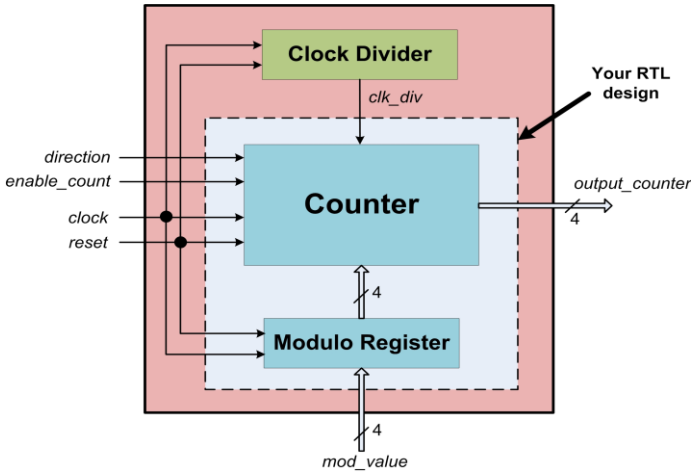

Figure 1: Block diagram.

clock	reset	enable_count	direction	mod_value	output_counter	*Description*
-	1	-	-	*Modulo Value*	0	Output will be zero and counter's upper limit sets to *modulo value*
↑	0	1	0	-	*Counter++*	Incremented by 1 where upper limit is *modulo value*
↑	0	1	1	-	*Counter--*	Decremented by 1 where upper limit is *modulo value*
↑	0	0	-	-	*Counter*	No change

Figure 2: Truth table.

The block diagram for the counter is given in Figure 1, and the truth table in Figure 2. The block diagram consists of three blocks:

1. Clock Divider : This block will produce a clock signal that pulses once per second. You are supplied with the VHDL code that does this (see Blackboard), and so you are only required to instantiate and use this block (and related clock signal) in your solution.

2. Counter : This is the heart of the counter. This block contains all the counting logic, including ensuring that the modulo value provided by Modulo Register is not exceeded, and produces the output signal. You are required to implement this block.

3. Modulo Register : This block is primarily storage—a register that stores the maximum value of the counter when the reset is activated. You are required to implement this block. You will need an internal signal (bus) that allows this block to communicate the value it stores to the Counter block.

4 Building a test bench

You will need to build, and implement a test bench so that you can test your design in simulation. You should consider all scenarios of input that you need to test so that you can ensure that your design is functionally correct. The testbench could use a combination of functional and regressive testing strategies to achieve an exhaustive testing of your digital system. When you are confident your design works correctly, you can load your design onto the development board.

5 Using the Cyclone III development board

Your solution should download and run on the DE0 board provided . Your logic will connect to the peripherals on the board that we have been using in this module, using the pin configuration file that we have been using since week 2 (supplied in the assignment folder). You should make an appropriate selection of input/output elements on the board to demonstrate your design, between SW, LEDG, BUTTON, HEX0, HEX1 … If you choose to use the HEX0, HEX1 7-segment displays you will need to include some logic for a BCD-7-segment display driver.

6 Documenting and reporting on your design

You are required to produce a short report (normally a few pages in length). In your report, you should explain how your design works, and any design decisions you made. For instance, you have been asked to choose whether a synchronous or asynchronous reset is appropriate. You should explain your choice (and the reason for your choice) here.

If there are errors in your design that you have found in simulation but not been able to debug, you can take the opportunity to explain them in your report. Credit will be awarded if you are able to demonstrate that your test bench either fully demonstrates that the design works, or that you have managed to expose an error and have some understanding of what the error is and why it may be arising.

To demonstrate the function of your design on the board you can optionally upload a video of your design working on the FPGA board to a video service (e.g. youtube) and include a link to the video in your report – do not upload a video file to Blackboard as the format may not be readable. The video should briefly show all the different behaviours with a short explanation.

7 Assesment submission requirements

Your submission will consist of 3 items:

1. Your VHDL source code for your design;
2. Your complete VHDL test bench;
3. Your report.

You are not required to submit compiled files such as bit files, or any of the files that you have been supplied with for this assignment. Your report should include timing diagrams generated from the behavioraral simulation in ModelSim.

You should submit your assignment electronically, via the blackboard site for this module (in the Assessment section, under Assessment 1 - A programmable 4 bit counter). Late submissions will be adjusted according to the university regulations. Incomplete submissions will only earn credit for those components that have been submitted.

8 Marking Breakdown

Grades for this assignment will depend equally on the quality of your VHDL code, technical achievement in the completeness of your test bench and its ability to exhaustively test the design, and the understanding that you demonstrate in your report.

Core hardware design using a combination of dataflow, structural and behavioural VHDL - 5%;
Test bench design VHDL 5%;
Design/Documentation/Test Report including demonstration of correct behaviour (docx/pdf) 15%.