ELEC2665 Unit 4 Assessment Brief 2021-2022

(May 2023)

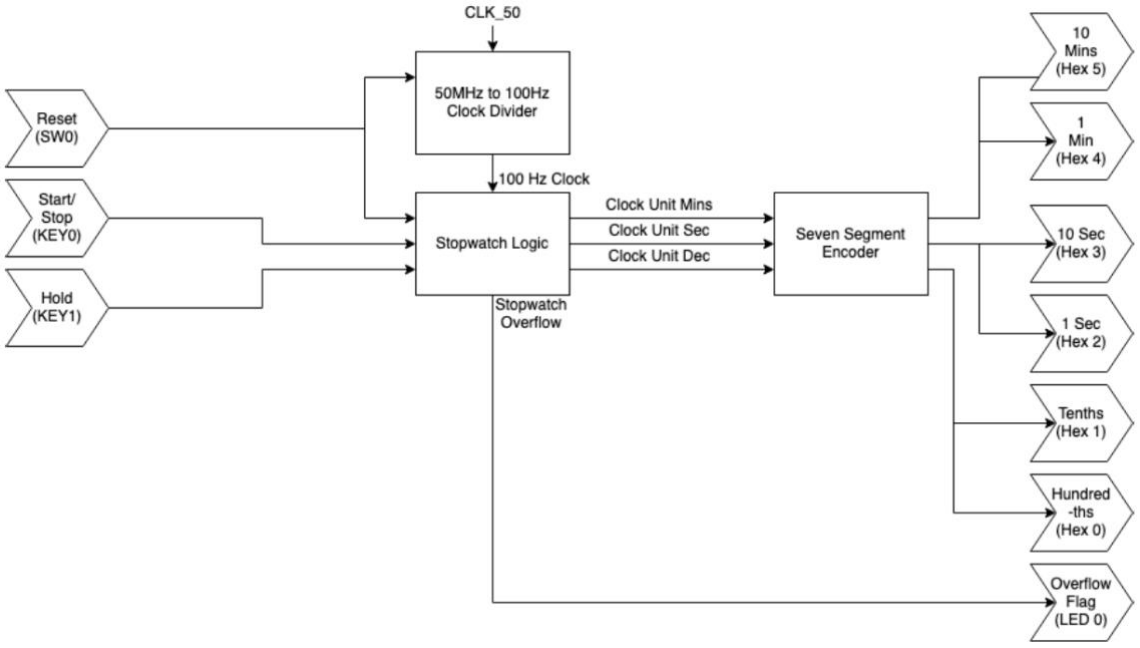
Stopwatch

Introduction

After the learning in Unit 4, I hope you all have got a further understanding of FPGA as well as practical experience in coding the FPGA board. So now, for the Unit 4 assessment, you will be invited to show your learning outcome by designing a Stopwatch based on the MAX10 FPGA architecture that you have been working with this semester. A template Quartus project that forms the foundation of your design is provided. It is up to you to edit and create new modules to achieve as much of the specified function as you can within the assessment time frame. In addition to the **practical work**, you will also be submitting a **design report** with which the template has been provided. You should answer all the sessions to the best of your ability, using diagrams and figures where appropriate to illustrate your answers.

Block Diagram

Here is the Block Diagram to illustrate the baseline structure of the Stopwatch.



System Design

Four modules should be implemented in this module:

Clock Divider – A clock divider that takes in a 50MHz clock signal generated by the MAX10 board and divides it down to 100Hz to drive the Timer Logic. Please refer to the Unit 4.3 screencasts for some ideas.

Stopwatch Logic – The main logic controller for the stopwatch, handling the counting of time and start/stop/reset functions. You will need to use your learning in FPGA sequential logic or structural logic for different functions.

Binary to Seven Segment Encoder – An encoder that takes the output from the logic controller and converts it to a format appropriate for display on four seven-segment displays. Though you have doneworklar works in Unit 4.2, you can try a different way for your implementation.

Stopwatch – The full system's top-level design, organised as shown in the block diagram.

Module Specifications

Clock Divider	
ClockDivider50MHzTo100Hz	
Inputs	
CLK_50MHz	A 50MHz clock signal from FPGA
reset_n	An active-low set signal
Output	
CLK_100Hz	A 1Hz clock signal
Function	
This module will divide down the 50MHz input clock into a 100Hz output clock, which will give a period of around 10ms for the timer logic. The reset will reset amd hold the output clock value to 0 when the reset signal is low.	

Stopwatch Logic	
StopwatchLogic	
Inputs	
CLK_100Hz	A 100Hz clock signal
reset_n	An active-low set signal
start_stop	An active-low start trigger
hold	An active-low hold singal
Output	
stopwatch_unit_mins [6:0]	The number of minutes [0-99]
stopwatch_unit_secs [5:0]	The number of minutes remains [0-59]
stopwatch_unit_decs [5:0]	The number of tenths of seconds elapsed on the timer (0-99)
Function	
A falling edge on the start/stop input should start the timer, which counts up until another falling edge is detected. A logic 0 on the hold input should pause the timer. A logic 1 will allow it to run. A falling edge on reset_n will reset the timer to zero and keep it there for as long as reset_n is low. The outputs should be split into buses representing minutes, seconds and tenths of seconds, with ranges as per the above. If the timer goes above its maximum range (99:59:99), the timer overflow signal should be set to logic 1.	
Binary to Seven Segment Encoder	

SevenSegEncoder	
Inputs	
stopwatch_unit_mins [6:0]	The number of minutes [0-99]
stopwatch_unit_secs [5:0]	The number of minutes remains [0-59]
stopwatch_unit_decs [5:0]	The number of tenths of seconds elapsed on the timer (0-99)
Output	
HexM_1 [6:0]	A signal encoded for 10 minute
HexM_2 [6:0]	A signal encoded for 1 minute
HexS_1 [6:0]	A signal encoded for 10 second
HexS_2 [6:0]	A signal encoded for 1 second
HexTS_1 [6:0]	A signal encoded for 1/10 second
HexTS_2 [6:0]	A signal encoded for 1/100 second
Function	
This module should take a binary input of minutes and seconds and encode these in a format appropriate for four seven-segment displays. For your convenience, a binary to BCD encoder has been included. However, that is not the best encoder in the world so please feel free to redesign it if you have additional time (of course, this will also bring you additional marks).	

Stopwatch	
Stopwatch	
Inputs	
CLK_50MHz	A 50MHz clock signal
reset_n	An active-low set signal
start_stop	An active-low start trigger
hold	An active-low hold singal
Output	
HexM_1 [6:0]	A signal encoded for 10 minute
HexM_2 [6:0]	A signal encoded for 1 minute
HexS_1 [6:0]	A signal encoded for 10 second
HexS_2 [6:0]	A signal encoded for 1 second
HexTS_1 [6:0]	A signal encoded for 1/10 second
HexTS_2 [6:0]	A signal encoded for 1/100 second
CLK_ind	Clock indication signal flash every 1 second
Overflow_flag	An active low signal to indicate timer overflow
Function	
This module should act as the top-level design for the Stopwatch as illustrated in the block diagram. The input and output signal should be connected to the following pins: CLK_50MHZ: MAX10_CLK1_50 HexM_1 HEX5[0-6] reset_n: SW0 HexM_2 HEX4[0-6] start_stop KEY0 HexS_1 HEX3[0-6] hold: KEY1 HexS_2 HEX2[0-6] CLK_ind HEX4[7] HexTS_1 HEX1[0-6] Overflow_flag LED0 HexTS_2 HEX0[0-6]	

Design Rules

- Try not to modify the existing document headers. You can add new input/output if you want to implement additional functions.
- **You need to pair All your modules with a testbench and validate your module with the testbench!**
- Follow the block diagram as a guide for the top-level design and the interconnection between main modules. This diagram act as a fundamental requirement for your project. You are encouraged to expand the function by adding your modules/function on top of it.
- I expect you to follow the design practices that have been covered this year. All main modules can and should be implemented with **primitive logic** and **counters**, with a couple of **behavioural** *if, Case* statements. That is to say, there are limitless ways (even within these bounds) to achieve the specified functionality.
- **It is indeed challenging to finish the whole thing!** Spend a reasonable amount of time on the practical work and do what you can to the best of your ability
- When we test your design, we will toggle (re)set and examine the functionality after that point – you will not be judged on anything that happens in between T0 and the reset signal.

Practical Submission

When you are ready to submit, compress your project folder (.zip files only!) and submit it via the link on Minerva. Ensure that you have included all of your submodule files and test benches (I would expect you to test **ALL** the modules with test benches), including those you may have reused from the taught material.

Report Submission

This time, a report template will be provided to you to guide your report writing. This template is designed based on the main contents and topics you will be expected to provide at the end of your individual and/or group project in Levels 3 and 4. You should finish each session fully and to the best of your ability. In each session, a detailed requirement is listed. Please make sure you read them carefully and cover all the contents asked. The page limit mentioned in each session is for guidance and will not be used for marking. Moreover, general tips in each session are also included for your reference. And the percentage number in each session also gives you a rough idea of how we expect you to distribute your discussion in each session.

The reports are to be submitted **via Turnitin** at the same time as your practical work.