

CSSE4000 / CSSE7002: Semester 2, 2006

Reconfigurable System-on-Chip Project

Introduction

This project explores the tradeoffs between software and hardware implementations of a simple System-on-Chip functionality.

Pulse Width Modulation (PWM) is a method of generating analog-like signals from purely digital circuits. By varying the duty cycle of a square wave signal from 0 to 100 percent, and filtering the output through a low-pass filter, analog outputs may be simulated.

You will use PWM to vary the brightness of the LED on the D2FT prototyping board. Varying the duty cycle of the LED control signal, and with your eye providing the low pass filtering, you can vary the apparent brightness of the LED from all-off to all-on.

Submission and Due Date

The assignment is due by the end of Semester 2, Week 13 (27th October). Marks will be deducted for late submissions according to ITEE Assessment Policy.

Completed assignments are to be in class (hardcopy report and CDROM. Electronic (PDF or DOC) versions of the reports are also to be submitted via the ITEE online submission system (<http://submit.itee.uq.edu.au>)

Group work

You may work *alone or in groups of two*.

If working in a group, each student must submit their own report, written in their own words.

Groups identify themselves to the lecturer no later than the lecture in Week 9 (September 18th). After this time all students not associated with a group will be assumed to be working alone.

Development boards and environment

The tasks are to be completed using the Xilinx ISE/EDK 8.1-version tools available in the labs.

Your systems are to be implemented on the Digilent D2FT FPGA boards, with the accompanying D5IO boards used to allow user-input via the buttons and switches.

Deliverables

Report (70% of marks)

Write a short report (5-10 pages of text/diagrams plus appendices) discussing the design and implementation of the tasks (see below), including:

- Introduction
- Design Overview, including
 - Blocks diagram of core internals
 - System diagrams showing core in contexts of whole system
 - System bus address maps
 - Any other explanatory material or text required to describe your design approach
- Results
 - Discussion of the results of your work
 - Synthesis reports (e.g. fragments of **system.par** files showing logic utilisation)
 - Simulation results (screen captures from simulation tools)
 - Show enough to prove functional status of the core
 - Bus interface signals, core outputs and significant internal signals
- Discussion
 - (see example discussion questions above, and think of some of your own)
- Appendix (not included in page count)
 - Commented source code files
 - MHS files for systems
 - VHDL files for custom core(s)
 - C source for MicroBlaze programs

Submit a printed copy of the report.

CD / electronic copy (20% of marks)

Submit a CD containing a *subdirectory for each of the tasks* (eg TaskA, TaskB, ...). Within each subdirectory, provide

- Complete, self-contained Xilinx EDK project. These will necessarily include
 - the system design files (e.g. XMP, MHS, MSS, UCF, ...)
 - custom core implementation (in the **pcores** subdirectory)
 - MicroBlaze C source code files(s).
- Final **system.bit** bitstream files, ready to download and test.

Also include an electronic copy (e.g. PDF) of your report.

Ensure your name and student number are written clearly on the CD in permanent ink.

Submit an electronic copy of your report via the ITEE online submission system (<http://submit.itee.uq.edu.au>)

Demonstration (10% of marks)

Each student or group shall demonstrate their work in the lab (max 5 minutes each). Demonstration sessions will be held in the prac session of Week 13 (Monday 23rd October) at the AMC facility. Students unable to attend this session must arrange an alternative demonstration time prior to this date.

Tasks

Task A – PWM Software implementation

Develop a MicroBlaze-based SoC that uses the standard OPB_GPIO core to drive the LED.

Write a MicroBlaze program that implements PWM to control the brightness of the LED.

The program shall allow the user to control the desired PWM brightness value by the buttons or switches on the D5IO board.

Task B – PWM Custom hardware implementation

Using the EDK “Create/Import Peripheral Wizard”, and implementing your own user logic, create a custom OPB core that uses 8-bit PWM to modulate the brightness of the LED.

Your custom core should have a single bus-mapped register that is used to set the PWM value (0-255).

Integrate your core into a MicroBlaze SoC, and write a MicroBlaze program to input the value from the D5IO board, and control the PWM output.

Task C – PWM Off-the-shelf IP core

The Xilinx standard OPB_TIMER core can be configured to operate in a PWM-generation mode. Re-implement the MicroBlaze LED PWM system using this Off-The-Shelf IP core to implement the PWM functionality.

TIP: The datasheets for Xilinx cores can be accessed from within the EDK / Platform Studio GUI. Right click on the core in the “System Assembly” view, and select “View Datasheet”.

Task D – Programmable Pulse Rate (*CSSE7002 students only*)

Copy and **modify your design from Task A, to pulse the PWM output smoothly from 0 to 100% and back down to 0 using purely software control of the output LED.** The pulse rate is to be specified by the user.

For example, a pulse rate of 0.5 seconds would raise the PWM amplitude from 0 to 100% in 0.25 seconds, then back down to zero in another 0.25 seconds (0.5 seconds total pulse period), and repeat.

You will probably need to add an OPB_TIMER peripheral to your system, to measure the passing of time.

The user must be able to specify the pulse rate using the button/switch inputs on the D5IO board.

Next, modify your core created in Task B so that the pulse rate is controlled in hardware, with the pulse rate set via an OPB bus-mapped register.

Once again, create a modified version of your controlling software from Task B to allow the user to specify the pulse rate.

Discussion topics

In your report, discuss the relative merits and weaknesses of the various implementations. Some questions to consider (think of your own as well!)

- How would each system change if two PWM channels were required? What about 256 channels, or 1024 channels? What implications does this have for HW vs SW implementations?
- Compare the relative complexity of the software required to implement the three solutions. What about the hardware?
- What would be the impact on the various implementations if the MicroBlaze CPU was required to perform some other software processing – for example MP3 decoding?
- Consider the logic and memory usage for each of the solutions – for example how does the logic usage of your custom core compare to the Xilinx GPIO implementation?