ECE 385

EXPERIMENT #7

SOC with NIOS II in SystemVerilog

I. OBJECTIVE

In this experiment you will learn the basic capability of the NIOS II processor as the foundation of your System-On-Chip (SOC) projects. You will learn the fundamentals of memory-mapped I/Os and implement a simple SoC interfacing with peripherals such as the on-board switches and LEDs.

II. INTRODUCTION

The goal of this lab is create a NIOS II based system on the Altera Cyclone IV device. The NIOS II is an IP based 32-bit CPU which can programmed using a high level language (in this class, we'll be using C). A typical use case scenario is to have the NIOS II be the system controller and handle tasks which do not need to be high performance (for example, user interface, data input and output) while an accelerator peripheral in the FPGA logic (designed using SystemVerilog) handles the high performance operations.

The Introduction to NIOSII and Qsys will give you a walkthrough of the Qsys wizard, which is used to instantiate IP blocks (including the NIOS II). We will set up a minimal NIOS II device with an SDRAM (Synchronous Dynamic RAM) controller and a PIO (Parallel I/O) block to blink some LEDs using a C program running on the NIOS II to confirm it is working. You will then be asked to write a program which reads 8-bit numbers from the switches on the DE2 board and sums into an accumulator, displaying the output using the green LEDs via the NIOS II. This will involve instantiating another PIO block to read data from the switches and modifying the C program to input data, add, and display the data.

Please read the **INTRODUCTION TO NIOS II AND QSYS** (INQ. 1-22).

Your top-level circuit should have **at least** the following inputs and outputs:
General Interface:

Inputs

KEY[0]	: logic	-- For Qsys-mapped hardware reset purposes
KEY[2]	: logic	-- For accumulator initialization ('Reset')
KEY[3]	: logic	-- For accumulator accumulation ('Accumulate')
CLOCK_50	: logic	-- 50 MHz clock input

```
LEDG        : logic [7:0]        -- LED display of the accumulator
SW          : logic [7:0]        -- Switches for the accumulation input
```

SDRAM Interface for Nios II Software:

Bidirectional ports (inout)

```
DRAM_DQ     : logic [31:0]   -- SDRAM Data 32 Bits
```

Outputs

```
DRAM_ADDR       : logic [12:0]   -- SDRAM Address 13 Bits
DRAM_BA         : logic [1:0]    -- SDRAM Bank Address 2 Bits
DRAM_DQM        : logic [3:0]    -- SDRAM Data Mast 4 Bits
DRAM_RAS_N      : logic;         -- SDRAM Row Address Strobe
DRAM_CAS_N      : logic;         -- SDRAM Column Address Strobe
DRAM_CKE        : logic;         -- SDRAM Clock Enable
DRAM_WE_N       : logic;         -- SDRAM Write Enable
DRAM_CS_N       : logic;         -- SDRAM Chip Select
DRAM_CLK        : logic;         -- SDRAM Clock
```

NOTE: For the partial credits, you may add LEDs, hex displays, switches, and/or buttons to the above lists.

III. PRE-LAB

A. Download the provided codes for Lab 7 on the ECE 385 course website. Follow the INQ tutorial to complete the NIOS II, memory, and the controller setup by performing the tasks as described in the tutorial. Your LEDG[0] should start blinking on your board as soon as the binary has been transmitted to the NIOS II CPU.

B. Modify the hardware and the software setup of the Lab 7 project to perform accumulation on the LED using the values from the switches as inputs. The green LEDs should always display the value of the accumulator in binary and the accumulator should be 0 on startup (all LEDs off). The accumulator should overflow at 255+1 to 0. (255 + 1 → 0, 255 + 2 → 1, etc.) Pressing 'Reset' (KEY[2]) at any time clears the accumulator to 0 and updates the display accordingly (turns all the LEDs off). Pressing 'Accumulate' (KEY[3]) loads the number represented by the switches into the CPU, adding it to the accumulator. The 8 right-most switches (SW [7:0]) are read as an 8-bit, unsigned, binary number with up being 1, down being 0. Push buttons should only react once to a single actuation.

C. Answer the *italicized* questions and fill in the table from the INQ tutorial. Be prepared to give answers to any of the questions from your TA when demoing. This is to ensure that you make an effort to research what the settings do instead of simply trying to "make the picture look like your screen".

Hints: Unit test the input and output. The output should already work, but make sure you can turn on and off every segment. If you have problems, check the schematic for the DE2, and make sure you are actually toggling the correct pins.

For this, and the rest of the class, you may use the C standard libraries (stdlib.h) or the C++ equivalents, this can save you a lot of work when coding in C.

You will need to bring the following to the lab:

1. Your code for the Lab. You can bring the code to the lab on a USB storage device, floppy disk, CD, FTP or using any other method.
2. Block diagram of your design with components, ports, and interconnections labeled.

IV. LAB

Follow the Lab 7 demo information on the course website.

V. POST-LAB

1.) Refer to the Design Resources and Statistics in IQT.30-32 and complete the following design statistics table.

LUT	
DSP	
Memory (BRAM)	
Flip-Flop	
Frequency	
Static Power	
Dynamic Power	
Total Power	

Document any problems you encountered and your solutions to them, and a short conclusion.

VI. REPORT

In your lab report, should hand in the following:

- An introduction;
- Written description of the operation of your circuit;
- Written purpose and operation of each module (include lab7_soc.v);
- Schematic block diagram with components, ports, and interconnections labeled;
- Answers to the *italicized* questions and fill in the table from the INQ tutorial;
- Answers to post-lab questions;
- A conclusion regarding what worked and what didn't, with explanations of any possible causes and the potential remedies.