ECE 178 - Embedded Systems
Assignment Four

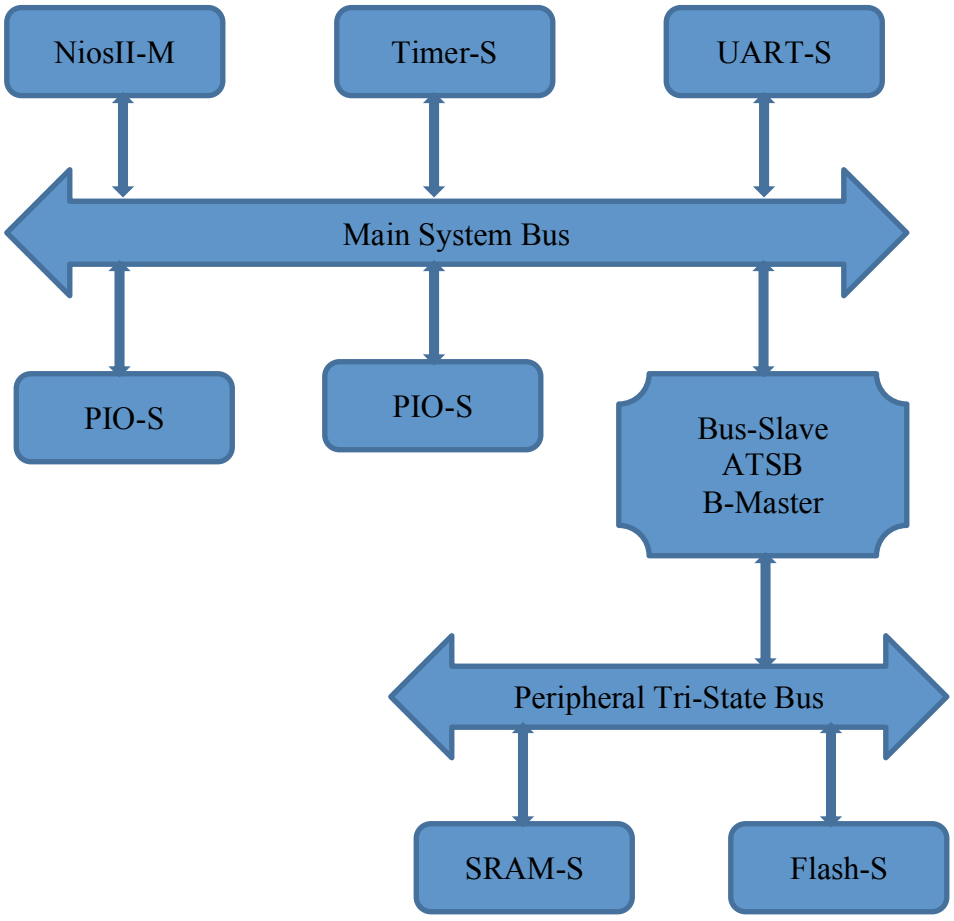
Hardware Design - Complete the Top-Level Schematic

After Qsys Builder has finished generating your embedded sub-system in previous assignment, open the niosII.bdf schematic in Quartus II by double-clicking on it on the project navigator. Add the niosII component to your Quartus II project by pressing OK. To complete the top-level schematic, add the necessary input, output, and bi-directional pins (and pin names) according to DE2 pin assignment file as shown in the class exercise.

Design Compilation

Verify that the pin assignments were made correctly by going to assignment →pins. A long list of pin numbers and names corresponding to the pin names you entered into the top-level schematic should appear. Then select Processing → start Compilation to begin compiling your project.

Hardware Design Architecture



The block diagram shown is an example of multiple external devices to share the same address and data bus pins for reducing the number of pins required on the FPGA. Your Nios II based design architecture could support similar bus sharing using tristate bus components. To accommodate the bidirectional data bus and multiple devices on a single bus, an Avalon Tristate Bridge component must be used. The Avalon Tristae Bridge creates a peripheral (tristate) bus to which multiple memory controller and other external components can be attached as shown.

Software Design – Application Development

Part I

Write a program that reads the contents of the switches, displays the corresponding value on the green LEDs, adds this number to a sum that is being accumulated, and displays the sum on the red LEDs and HEX displays.

Part II

In this part, we will use the polling approach to read the numbers entered via the toggle switches. The desired operation is that the user provides the next number by setting the toggle switches accordingly and then pressing the pushbutton *KEY*3 to indicate that the number is ready for reading.

To accomplish this task it is necessary to implement a mechanism that monitors the status of the circuit used to input the numbers. A commonly-used I/O scheme, known as *polling*, is to use a *status flag* which is originally cleared to 0. This flag is then set to 1 as soon as the I/O device interface is ready for the next data transfer. Upon transferring the data, the flag is again cleared to 0. Thus, the processor can *poll* the status flag to determine when an I/O data transfer should be made.

In our case, the I/O device is the user who manually sets the toggle switches and presses the pushbutton key. The I/O interface that provides the desired control is the one-bit status-flag PIO circuit generated, which includes the edge-capture capability.

Perform the following:

1. Modify your application program to accept a new number when the pushbutton *KEY*3 is pressed. This action will set the status-flag bit in the *edge-capture* register to 1. After reading the new number, your program has to clear the flag by writing a 0 into the *edge-capture* register.
2. Download and run your program to demonstrate that it works properly. The program should run continuously and a new number should be added each time the pushbutton *KEY*3 is pressed.

Part III

Instead of using the polling approach to read new numbers from the toggle switches, we now want to use interrupts for the same purpose. To accomplish this, we will use the ability of the status-flag PIO to raise an interrupt when the pushbutton *KEY*3 is pressed.

Modify your assembly language program to realize the desired task by using the interrupt approach.

Preparation

Your preparation should include the following:

1. Hardware System Design
2. Software Design and Assembly language application programs