Introduction

In Lab7, we create a basic SoC based on the NIOS II system on the Altera Cyclone IV. The simple system allow the FPGA compile and execute C code with parallel input and output system. With the basic SoC, we blink the LEDs on the board and develop an accumulator in C code, that can keep adding numbers until it overflows.

Written Description and Diagram of NIOS II System

The hardware component of this lab is an NIOS II processor, and a SDRAM. In the lab, we connect the processor with the controller of the SDRAM. The controller can help decide the dimension of the SRAM on the chip, and the phase shifter provide the SDRAM with correct clock that the IO can synchronous with the processor. For the PIO part, we create parallel IO system for the LEDs and the Switches to display and modify the C program to input data, add and display them with faster speed.

The Software components are below:

1. The Blinker of the LED
   1. First, we define the actual address of the LEDPIO to connect the software with the hardware. Second, we clear the LED by writing 0s to the register. Then we create infinite loops to keep the LED blinking. In the loop, we will let LEDPIO |= 1 loop for several to make the LED memory to and b’1 to high the LSB for some time, then we will keep the LEDPIO &= 0 to make the LED memory and b’0 to lower the LSB.
2. The Accumulator:
   1. First, we define the actual address to the pointers that the program can access and directly write value to the actual hardware. Then we clear the LED and the values that record the status of the bottom and the value of the accumulator. Then we create an infinite loop to keep the program running. In the loop, we check if the Reset bottom is pressed. If so, the program will reset the LED and the value of the accumulator, and change the status of the reset bottom inside the program until it is released. Then we check if the accumulate bottom is pressed. If so, we will add the value in the switches with the stored value. If overflow happened, it will loop back to smaller value. Then the status of the accumulated bottom will return to 0 when the bottom is released. Finally at the end of the loop we will display the current value of the accumulator on the LED.

Top level block diagram

Written Description of all .sv Module.

Module: lab7

Inputs: Clock\_50, [3:0] KEY, [7:0] SW

Output: [7:0] LEDG, [12:0] DRAM\_ADDR, [1:0] DRAM\_BA, [31:0] DRAM\_DQ, [3:0] DRAM\_DQM, DRAM\_CAS\_N, DRAM\_CKE, DRAM\_CS\_N, DRAM\_RAS\_N, DRAM\_WE\_N, DRAM\_CLK.

Description: The top-level file of LAB7.

Purpose: The top-level file declares, contents, and connects all the department together. For SoC, the top-level connects it with the hardware like LEDs, Switches and SDRAM, and connect the DRAM with the board memory.

Module: lab7\_soc

Input: clk\_clk, acc\_reset\_wire\_export, acc\_wire\_export, reset\_reset\_n, [7:0]sw\_wire\_export.

Output: [7:0] led\_wire\_export, reset\_reset\_n, sdram\_clk\_clk, [12:0] sdram\_wire\_addr, [1:0] sdram\_wire\_ba, sdram\_wire\_cas\_n, sdram\_wire\_cke, sdram\_wire\_cs\_n, [31:0] sdram\_wire\_dq, [3:0] sdram\_wire\_dqm, sdram\_wire\_ras\_n, sdram\_wire\_we\_n

Description: This is the SoC module that generated by the Platform Designer.

Purpose: The SoC module contents all the hardware information and the connection. It is generated from the PD based on what we done with the UI system.

System Level Block Diagram.

Post-lab Question

Conclusion

1. Our design transforms the FPGA board into a simple MCU that it can compile and execute simple C software. During the lab, we meet Netlist error when we try to compile the project. This was cause by several undefined interfaces during the design. Next time we could check them twice before exit the design.
2. The lab manual is pretty good since it proves detailed steps for new learners who touch the SoC at the first time. However, I think you can bookmark the lab manual to make it easier to look up when we meet some problems.