Education Objective

The educational objective of this laboratory is to investigate the use of interrupts from I/O devices interfacing with the NIOS II processor.

Technical Objective

The technical objective of this laboratory is to design an embedded system for the Nios II processor and DE1-SoC development board that will execute three different C programs that do the following:

1. Uses polling to increment/decrement a counter when KEY1 is pushed
2. Uses interrupts to increment/decrement a counter when KEY1 is pushed
3. Maintains the functionality of #2 in addition to flashing the LEDs every 100 ms using a timer interrupt

Demonstration Procedure

Part 1 - Hardware

1. Open Quartus II and create a new project
2. Open tools > QSYS
3. Create a system with the following components
    a. Nios II/e processor
    b. On-chip memory for program code and data
    c. 8-bit input PIO for switches
    d. 4-bit input PIO for pushbuttons
    e. 8-bit output PIO for LEDs
    f. 7-bit output PIO for hex0
    g. JTAG Uart
    h. Sysid
    i. Interval timer set for “simpler periodic interrupt” and a timeout period of 100 ms
4. Double click on the 4-bit PIO to open the configuration popup. To enable a pushbutton interrupt, check synchronous capture for the edge capture register, check Generate IRQ and choose EDGE for the IRQ type.
5. Verify that the timer, the jtag_uart and the pushbutton PIO are all connected to the NIOS II irq port. Edit the priorities so that the timer has highest priority and the jtag_uart has lowest priority.
6. Save your system as nios_system.qsys
7. Generate the VHDL
8. Return to the Quartus project and add nios_system.qip to the project
9. Create the top_level VHDL file. The <project>/nios_system/nios_sytem_inst.vhd file contains the component declaration and the port map template to instantiate nios_system in the top_level.
10. Use Assignments > Import Assignments… to import the pin assignments in the DE1_SoC.qsf file
11. Compile the design
12. Program the DE1_SoC board.

Part 2 – Software

1. Write a NIOS II C program that does the following:
    a. Displays 0 on hex0
    b. Checks to see if key1 is pushed (active low)
        i. If SW0 is high, increments the value on hex0
        ii. If SW0 is low, decrements the value on hex0
    c. Do not increment or decrement the value on hex0 until key1 is released
2. Open NIOS II Software Build Tools for Eclipse. Create a new App and BSP, generate the bsp, copy system.h to the app folder, move your C program to the app folder, build the project and choose Debug as NIOS II hardware. Click the run icon and verify your program works as expected.
3. Obtain signoff.

4. Write a NIOS II C program that does the following:
    a. Registers and enables a pushbutton interrupt for key1
    b. Displays 0 on led0
    c. Contains a pushbutton ISR that checks the state of SWO
        i. If SW0 is high, it increments the value on hex0
        ii. If SW0 is low, it decrements the value on hex0
5. Remove the first C program from the app folder and add the second one. Generate the bsp, copy system.h to the app folder, move your C program to the app folder, build the project and choose Debug as NIOS II hardware. Click the run icon and verify your program works as expected.
6. Obtain signoff.

7. Write a NIOS II C program that does the following:
    a. Maintains the functionality of the second program
    b. Registers a timer interrupt
    c. Contains a timer ISR that toggles the state of the 8 LEDs each time it is reached.
8. Remove the second C program from the app folder and add the third one. Generate the bsp, copy system.h to the app folder, move your C program to the app folder, build the project and choose Debug as NIOS II hardware. Click the run icon and verify your program works as expected.
9. Obtain signoff.