LAB Assignment #2 for ECE 522

Assigned: Wed., Sept 16, 2015
Due: Wed., Sept. 23, 2015

Description: Compile the Linux kernel with Soft GPIO support, update the device tree, implement lab1 functionality with a device driver (delete the 'mmap' function in the starter C code')

1) Vivado component: See lab 1: You will do the same thing, e.g., a block diagram with 1 instance of a GPIO with the first channel connected to the switches and the second channel set as 'custom' and labeled 'full_adder_inputs'.

2) Linux component: Run 'menuconfig', select 'Device Drivers', 'GPIO Support', 'Xilinx GPIO support', hit the space bar to add a '*' (note: 'Xilinx GPIO PS', the entry directly below 'Xilinx GPIO support' should already be selected). Save changes to generate a .config file. Compile kernel.

3) Add the following to the devicetree.dts directly below the 'ps7_gpio_0: ps7-gpio@e000a000' component (use the devicetree.dts file given on my website):

```
gpio: gpio@41200000 {
    #gpio-cells = <2>;
    compatible = "xlnx,xps-gpio-1.00.a";
    gpio-controller ;
    interrupt-parent = <&ps7_scugic_0>;
    interrupts = < 0 20 4 >;
    reg = < 0x41200000 0x10000 >;
    xlnx,all-inputs = <0x0>;
    xlnx,all-inputs-2 = <0x0>;
    xlnx,dout-default = <0x0>;
    xlnx,dout-default-2 = <0x0>;
    xlnx,gpio-width = <0x32>;
    xlnx,gpio2-width = <0x32>;
    xlnx,interrupt-present = <0x1>;
    xlnx,is-dual = <0x1>;
    xlnx,tri-default = <0xffffffff>;
    xlnx,tri-default-2 = <0xffffffff>;
} ;
```

4) Compile the device tree into the devicetree.dtb file. Copy the zImage and devicetree.dtb file to SD card.

5) Boot the system: NOTE: You MUST stop the boot process at the zynq-boot prompt, program the FPGA and THEN continue the boot process (type 'boot' at the zynq-boot prompt).

6) Once booted, you can create device nodes for the new device and control it using the following commands from the command prompt:

Input channels from Switches:

```
echo 206 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio206/direction
cat < /sys/class/gpio/gpio206/value
```

Note: The number ‘206’ may be different for you. Simply do an ‘ls /sys/class/gpio’ to see the controllers, which will be given as ‘gpiochipxxx’ where xxx can be 206. Note that the MIO and EMIO GPIO controller chip is always listed as ‘gpiochip0’. Do not use 0 as the number above since you did not connect the LEDs to the EMIO channels.

Output channels to LEDs:

```
echo 206 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio206/direction
echo 1 > /sys/class/gpio/gpio206/value
```

7) Update the ‘GPIO_starter_code.c’ to create the device nodes, open the device node and then add a loop that carries out a ‘read’ on the input channel and a ‘write’ on the output channel. Note that you’ll need to use ‘lseek’ to return the file pointer to position 0 after each read.

Laboratory Report Requirements:

Grading:
The grading from this lab will be based entirely on your in-class demo. Bonus points will be given to any implementation feature that goes above and beyond the requirements.