ECE4530 Fall 2017
Homework 6: Introduction to ARM programming on the FPGA-SoC

Assignment posted on 26 October
Solutions due on 2 November 5PM

Homework 6 is an introduction to programming the ARM in the FPGA-SoC of the DE1-SoC FPGA board. This Homework walks you through the steps of setting up the board to boot Linux, compiling a test program, downloading it onto the board, and running it. You will try to estimate, as accurately as possible, the execution time (in clock cycles) of a given C function. Next, you will explore optimization techniques for software.

To complete this homework, you will need the following software and hardware, in addition to the DE1-SoC board and the Altera SoC EDS software:

- A Micro-SD card of 4GB or more. Unfortunately these cards are not included with the DE1-SoC kit. If you do not own one, you'll need to purchase it. They cost about $5, and are easy to find (VT Bookstore, Bestbuy, Target, Walmart, Amazon, ..). The smallest size (4G) will be adequate for this course.

- A disk image writing program. The instructions in this homework are given for Win32 Disk Imager, which can be downloaded from `https://sourceforge.net/projects/win32diskimager/`

- A terminal program. The instructions in this homework are given for MobaXterm, which can be downloaded from `http://mobaxterm.mobatek.net/`.

- An Ethernet cable. This will be used to network your board.

- A USB cable with USB mini-B connector on one side, and a standard USB A connector on the other side. This will be used to connect the terminal application on your laptop to the USB UART on the board.

- A copy of the Linux SD card system image. You can download this from `https://canvas.vt.edu/files/5186485/download?download_frd=1`.

Initial hardware and software installation

1. Mount the MicroSD card on your laptop. Typically, this is done by mounting the microSD card into an SD card carrier. If your laptop has an SD card slot, then you can insert the SD card carrier directly. Alternately, you can use an SD-to-USB convertor that will fit into a USB slot of your laptop, and that mounts the SD card as a USB drive.

2. Install the disk image writer program on your laptop, and download the Linux SD card image from the course repository. The archive expands to a 1.8G image file `DE1_SoC_SD.img`.

3. Start the disk image writer program and select the `DE1_SoC_SD.img` file as the source image, and the SD card as the target drive. Pay extra attention during this step. **If you select the wrong target drive, you could corrupt the hard drive of your laptop.** The Win32 Disk Imager program is usually clever enough to point out the correct drive letter for you, but I am convinced that you are smarter. Double check it.

4. Connect the USB mini-B connector to your DE1-SoC kit. The connector is in the upper right corner of the board, at the side of the Ethernet connector. Connect the other end of the USB cable to your laptop.

5. Connect the Ethernet connector into your board. Connect the other end to your router. I'm assuming that you have home router to do this - but there are some alternatives, such as using a Raspberry-PI or a direct connection to your laptop. Contact the TA or instructor if a router is unavailable to you.

6. Insert the micro SD card into the micro SD card slot of the DE1-SoC board.

7. Install and start the terminal application. Open a serial port connection to the board. If you are using MobaXterm, click Session - Serial and look for a 'USB Serial Port' in the pulldown menu. The board has to be turned on to see this port. In the Speed pulldown menu, select '115200' baud. In Advanced Serial Settings, select 8 data bits, 1 stop bit, no parity, and NO flow control ('None'). Click OK.

8. Since you have already turned on the board, it has already booted. If you press return, you should see the following prompt in the terminal:

```
Poky 8.0 (Yocto Project 1.3 Reference Distro) 1.3
 ttyS0

socfpga login:
```

   If you don't see anything happening, verify that you have a connection to the board. Each time you press Enter when the cursor is positioned in the terminal window, you should see a blue LED blinking on the board, near the mini-B USB connector. You can also reset the board by pressing the 'warm reset' or 'HPS reset' buttons on the lower-right corner of the board. Pressing these buttons will restart the boot sequence.

9. When you log in as `root` you will see a prompt and you are logged into the board. Try some linux commands like `dmesg` and `cat /proc/cpuinfo`.

10. Give your board a root password using the `passwd` command. The initial password is empty.

Configuring the Network

To compile ARM binary code, and FPGA bitstreams to your board, we will use an Ethernet connection, and a secure shell. You can use the serial terminal in MobaXterm to configure the network settings of your DE1SoC as follows.

1. First, figure out what network you are connecting to. Home routers typically have network IP addresses of the form `192.168.0.x`, so that is the assumption made in the following instructions. You also need to know the IP address of the router that you connect to using the Ethernet cable. By default this is `192.168.0.1`.

2. You can double check your home wireless network ip address by typing `ipconfig` at a Cygwin prompt, when you are connected to that wireless network. You can also check the address of the router by looking for the 'gateway' line in the output of `ipconfig`.

   For example, on my home network, `ipconfig` shows:

   ```
   Wireless LAN adapter Wi-Fi:

      Connection-specific DNS Suffix  . :
      Link-local IPv6 Address . . . . . : fe80::c0ef:3d1d:33d1:6adb%14
      IPv4 Address. . . . . . . . . . . : 192.168.0.117
      Subnet Mask . . . . . . . . . . . : 255.255.255.0
      Default Gateway . . . . . . . . . : 192.168.0.1
   ```

   The router is at `192.168.0.1` and IP addresses are of the form `192.168.0.x`

3. Now you need to choose an unused IP in your network. The following instructions are for a fixed, static IP for your DE1SoC board. This is convenient to configure MobaXterm to log in to your board. Pick an unused IP address on your network, for example `192.168.0.55`. You can `ping 192.168.0.55` to check if nothing is already using that address.

4. Use MobaXterm and a serial terminal to log in as root to the board. Go to the `/etc/network` directory and open the file `interfaces` in an editor (for example, `vi`). Add the following lines to `interfaces`, replacing any lines related to `eth0`:

   ```
   # Wired or wireless interfaces
   auto eth0
   iface eth0 inet static
   address 192.168.0.55
   netmask 255.255.255.0
   gateway 192.168.0.1
   ```

If you have never used `vi`, then there is an alternative by using cut-and-paste from the MobaXterm screen, using a windows editor, and finally pasting back the file on the command line using `cat >interfaces`. But, it's faster to try `vi`.

5. When you have made the change, type `reboot` on the command line to reboot the board. When the connection is back, test it from your Cygwin using `ping 192.168.0.55`. If all goes well, your board will respond.

6. Finally, you can set of the SSH connection. In MobaXterm, open a new SSH session, using the IP address of your board. Use `root` as username. When you connect to the board, you should see a prompt:

   ```
   root@192.168.0.55's password:
   ```

   which means that the connection is successful. Having your DE1SoC board attached to the network will make your life much easier.

Hello World

Once you have correctly booted and networked the board, you are ready to write software for it and run it. You can check out the examples for Lecture 13, for example. The following sequence of commands compile the helloworld example and copy it to the board.

```
git clone https://github.com/vt-ece4530-f17/lecture13-101717-code
cd helloworld
make
scp hello root@192.168.0.55:/home/root
```

In an SSH terminal on the board, check if the program has copied correctly, and run it.

```
root@socfpga:~# ls
hello
root@socfpga:~# ./hello
Hello, World
root@socfpga:~#
```

The assignment

Once you have completed the above setup, you are ready for the assignment. In the repository of the homework, you will find three directories.

- `cyclecount`: Same as from `Lecture13`, this example illustrates how to count clock cycles.

- `hexspeed`: The main directory for the assigmment.

Perform the following commands.

1. Go to the directory `hexspeed/hardware`
2. Copy the three files you find there to the board.

   ```
   scp hps_config_fpga root@192.168.0.55:/home/root
   scp HPS_LED_HEX root@192.168.0.55:/home/root
   scp soc_system_dc.rbf root@192.168.0.55:/home/root
   ```

3. Configure the FPGA by running the following command on the board terminal. The following shows the command and its output; you only have to type the command at the commmand line.

   ```
   root@socfpga:~# ./hps_config_fpga soc_system_dc.rbf
   INFO: alt_fpga_control_enable().
   INFO: alt_fpga_control_enable OK.
   alt_fpga_control_enable OK  next config the fpga
   INFO: MSEL configured correctly for FPGA image.
   soc_system_dc.rbf file file open success
   INFO: FPGA Image binary at 0x72c36008.
   INFO: FPGA Image size is 2248310 bytes.
   INFO: alt_fpga_configure() successful on the 1 of 5 retry(s).
   INFO: alt_fpga_control_disable().
   ```

   Once the command completes, you see the HEX display of the board light up.

4. Now, you can run the software application that uses the HEX display.

   ```
   root@socfpga:~# ./HPS_LED_HEX
   hex show 0
   LED ON
   hex show 1
   hex show 2
   LED OFF
   hex show 3
   ...
   ```

5. The previous commands illustrate two things: (1) The ARM can download a bitstream into the FPGA. (2) The ARM software can communicate with the hardware configured in the FPGA. The hardware configuration used here consists of a simple PIO port that drives the HEX display.

6. Now move to the directory `hexspeed`. Compile the code, and download it to the board. Run it. You will see the HEX display counting very fast until `0xFFFFFE`, and then end. If you inspect `main.c` of the `hexspeed` program, you'll find the following main loop:

```
for (i = 0; i<0xFFFFFF; i++) {
  printhex(i);
  printhexmem(i);
}
```

7. Compare the source of `printhex` and `printhexmem`. Both of them are nearly identical, but they write to a different memory address. `printhex` uses calls to `alt_write_word` to write directly to the PIO on the FPGA. `printhexmem` writes directly into a global variable called `memhex`.

8. So finally, 6 pages in, we're able to formulate the main question for this Homework: **Find out how much faster printhexmem is compared to printhex?**. You can make modifications to `main.c` of hexspeed to make this measurement. You may make use of the code in `cyclecount` to count the clock cycles during the execution of `printhex` and `printhexmem`.

9. The final output of your modified `hexspeed` program should have the following output. It does not have to match identically, the output shown, as long as it has all of the information indicated.

```
printhex uses a median of XXX cycles per call
printhexmem uses a median of YYY cycles per call
```

   You would replace XXX and YYY with the actual measurements made by your program. Keep in mind that the `hexspeed` program can only run when you have configured the bitstream (after power-up of the board). So if you turn off the board, you will have to execute `./hps_config_fpga soc_system_dc.rbf` again to configure the bitstream, before you can run `hexspeed`.

10. That's it!

What to turn in

The homework is due before 2 November, 5PM.

1. You have to push a modified `hexspeed/main.c` from `hexspeed` into your repository.