CRASH COURSE ON PTARM (ON XILINX)

Contributors to the PRET project in general are (in alphabetical order):
David Broman, Dai Bui, Stephen A. Edwards, Sungjun Kim, Matthew Kuo, Edward A. Lee, Ben Lickly, Isaac Liu, Hiren D. Patel, Jan Reineke, Martin Schoeberl, Eugene Yip, and Michael Zimmer.

Document created: Monday, 14 October 2013
Document last modified: Friday, 24 June 2016

Contents

Preliminaries

Project Directory Structure

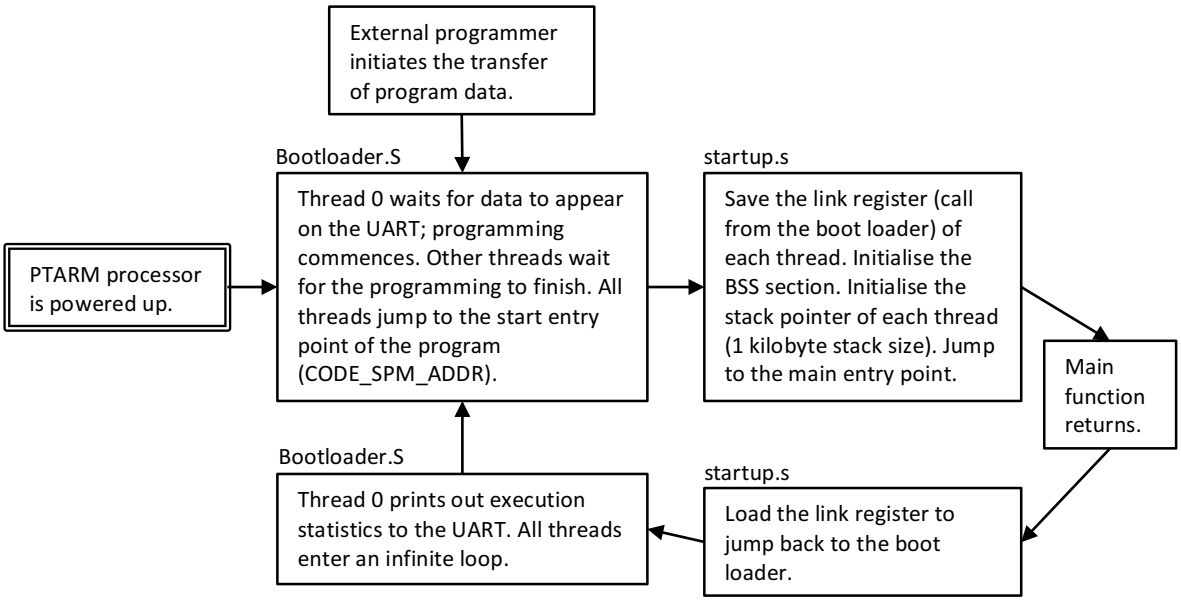
The diagram below describes the important folders needed to synthesis and program the PTARM processor.

PTARM

- **boot**
  - PTARM boot loader ASM program that forms the PTARM boot ROM.
- **compiler**
  - Compatible GCC ARM compiler
- **de2-115**
  - Main Altera Quartus project for PTARM.
- **doc**
  - Documentation.
- **include**
  - C header files for writing C programs on PTARM.
- **programs**
  - Test programs.
- **scripts**
  - Compiles and runs C programs on PTARM.
- **syn**
  - Synthesisable VHDL files for PTARM.

Understanding the PTARM Processor

Precision-Timed ARM (PTARM) is a VHDL implementation of the Berkeley PRET microarchitecture. The current PTARM implementation supports ARMv2, extended with timing instructions. **Isaac Liu**'s thesis on "**Precision Timed Machines**" should be read to understand what the PTARM processor can achieve. This crash course document assumes that the reader is already familiar with PTARM and wishes to fix, modify, or extend the design. The **architecture-overview.ppt** document in the **doc/architecture/** folder gives a schematic overview of the VHDL design. The diagram below summarises the life cycle of a program:



Building the PTARM Processor

The following procedures have been tested on Windows 10 + Quartus Prime 15.1.0 Lite Edition + Altera DE2-115 Development and Education Board + GCC 5.2.1:

1. Check out the source code.
2. Start up Quartus Prime and load the project file in **de2-115/PTARM.qpf**
3. In the **Tasks** panel, double click **Compile Design**.
4. In the **Tasks** panel, double click **Program Device** to open the **Programmer** window.
5. Perform a **Hardware Setup** for the JTAG.
6. Add the compiled SOF that was generated by Quartus Prime using the **Add File…** button.
7. Set the Altera DE2 board to Normal Mode (RUN) using the slide switch next to the LCD.
8. Click **Start** to program the Altera DE2 board.
9. On the Altera DE2 board, press the **KEY0** (CPU reset).
10. The serial port should send a string detailing the PTARM version and modification date.

Compiling Programs for the PTARM Processor

In the **scripts** folder, there is a Python script called **ptarm** that is used to compile, download, and execute a C program on the PTARM processor. The script communicates with PTARM using UART. A PTARM compatible ARM compiler can be found here: https://launchpad.net/gcc-arm-embedded.

To clean: `$ ./ptarm clean`
To create: `$ ./ptarm create`
To compile: `$ ./ptarm make`
To execute: `$ ./ptarm execute`
To clean then create, compile, and execute: `$ ./ptarm run`

The script contains templates for creating the program start up code, linker script, and makefile. Check that the **PTARM_PATH** in the script points to the correct **include** folder. There are test programs in the **programs** folder.

Customizing the PTARM Processor

Changing the Size of the PTARM Scratchpad

Change the **SPMPOWERSIZE** parameter (**line 12**) in **arm_defs.vhdl** (in the **syn/core/** folder). The **THREADBITS** constant is not used explicitly in the SPM component because the hardware threads access the same memory space. Next, follow the instructions for modifying the memory layout.

Modifying the Boot Loader

To bring the PTARM processor to a minimum working state on start up, the PTARM processor includes a small boot ROM. The program counter of each PTARM hardware thread begins at 0xFFFE 0000. Once the program has been loaded into memory, the PTARM hardware threads set their program counter to 0x000 0000. The PTARM boot loader (**bootloader.S**) is found in the boot **folder**. The boot loader is integrated into the PTARM VHDL design files as a memory array. To regenerate the boot ROM, execute the **Makefile**.

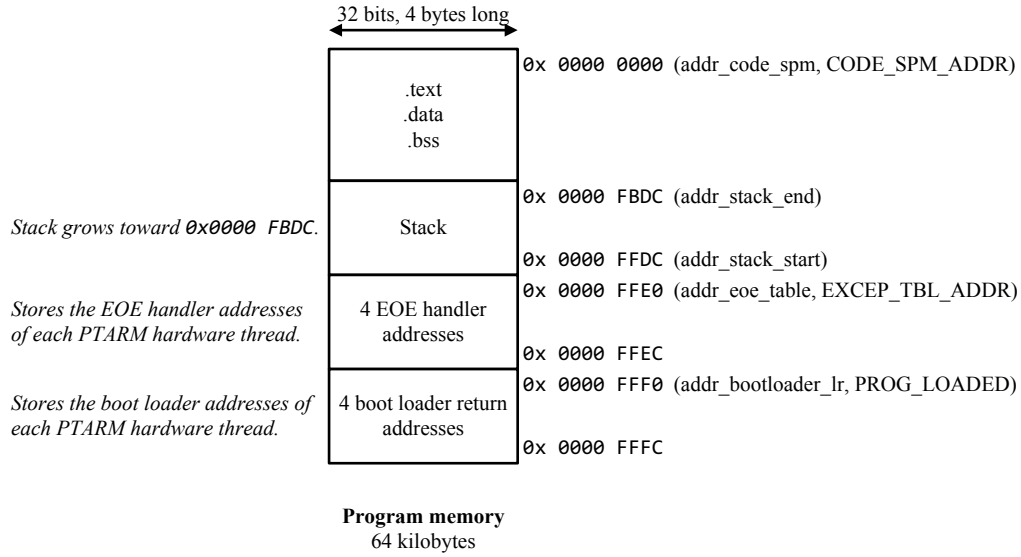
Loading the Programmer Object File onto On-Board Flash Memory

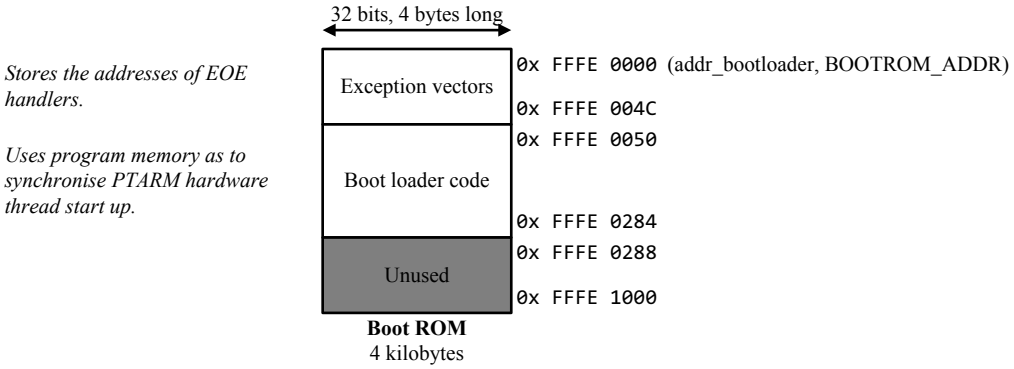
The Altera DE2-115 board includes on-board flash memory that can be used to store non-volatile data. The memories can be configured to be read when the board is powered on. Loading the PTARM programmer object file (POF) into flash memory allows the PTARM design to be automatically configured into the FPGA on power up, eliminating the need to manually configure the FPGA via JTAG.

1. Convert the SRAM Object File (SOF) generated by Quartus Prime using the **Programmer** tool.
    a. Open the Programmer tool from the Windows Start menu, not from within the Quartus Prime application.
    b. File > Convert Programming Files.
    c. In the window that opens, select the correct **Configuration device** (EPCS64).
    d. Provide the correct POF file name.
    e. Select **SOF Data** in the **Input Files to convert** table.
    f. Add the SOF using the **Add File…** button.
    g. Click **Generate** to generate the POF.
2. In the **Programmer** tool, change the mode from JTAG to **Active Serial Programming**.
3. Add the POF using the **Add File…** button.
4. Set the Altera DE2 board to AS Mode (PROG) using the slide switch next to the LCD.
5. Click **Start** to program the Altera DE2 board.

Modifying the Memory Layout

PTARM uses the following memory address spaces for software.



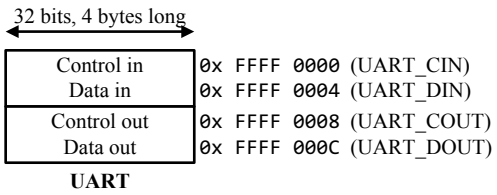


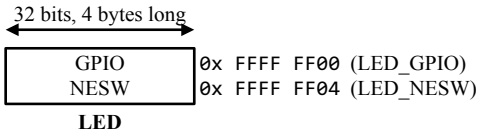
If you want to change the location of the Exception On Expire (EOE) handler table, then you need to change the value of **EXCEP_TBL_ADDR** in **bootloader.S** (in the **boot** folder) and the value of **addr_eoe_table** in the **ptarm** script (in the **scripts** folder). Remember to regenerate the boot loader.

If you want to change the start address of the program, then you need to change the value of **CODE_SPM_ADDR** in **bootloader.S** (in the **boot** folder) and the value of **addr_code_spm** in the **ptarm** script (in the **scripts** folder). Remember to regenerate the boot loader.

If you want to change the start address of the stack, then you need to change the value **addr_stack_start** in the **ptarm** script (in the **scripts** folder).

PTARM uses the following memory address spaces for memory mapped inputs/outputs.





Check the Boot ROM source code for the actually memory mapped input/output types and addresses.

Checking the Memory Layout

To check that the physical amount of synthesised memory matches the intended amount, a memory testing program can be used. If writing to an address and reading from the same address gives an unexpected value, then the address is probably not addressable. An example program (**memory**) that tests a range of memory addresses can be found in the **test** folder.

Using the Timer, Mutex, and Divider

The timer, mutex, and divider are implemented as part of the PTARM datapath, rather than as separate coprocessors. The timer and mutex are accessed by the coprocessor instruction but share the general purpose registers of the executing hardware thread. Thus, coprocessor data transfers are not needed.