Lab 3.0
Hybrid Systems

Lab 2.0 + 2.1 – Thermal Camera & Full-System Integration

The goal of the lab 2 series was to design a specialized component in order to interface with the Lepton thermal camera. You built 2 pieces of this interface, then created an *instantiable component* that represents it within a *system integration tool* (Qsys). You then used the tool to compose a full system that captures images from the Lepton and saves the output to a file on your host machine.

As such, you create a system that runs *exclusively* on the programmed *FPGA fabric*.

Lab 3.0 – Hybrid Systems

Introduction

The Cyclone V SoC architecture is a hybrid design: it does not solely consist of an FPGA, but rather of a single chip containing both a HPS[1] *and* an FPGA. The HPS consists of an ARM-A9 MPCore, which is a general-purpose *application processor*. In this lab, we will explore using the HPS instead of the softcore Nios II processor you have used until now.

Although the HPS was designed to run an operating system, you still lack familiarity with the tools needed to create such a system. We will come back to this next week in lab 3.1. For the moment, you are instead going to use the HPS to run bare-metal code, as the setup is much shorter and gives you more time to get acquainted with the tools.

Getting Started

Getting the HPS up and running is a much more involved process compared to the *embedded* Nios II processor, which is to be expected since it is a *general-purpose* processor.

We would normally tell you to **RTFM**, but given the huge search space of documentation available, we have written a step-by-step tutorial for getting up to speed with development for Cyclone V SoC-based devices. You can follow the tutorial by reading the SoC-FPGA Design Guide.

[1] *Hard* Processor System (a.k.a not synthesized on the FPGA fabric, but a "real" processor)

René Beuchat, Philémon Favrod, Sahand Kashani

The tutorial was written for the “bare” DE0-Nano-SoC board (without the PrSoC extension board). However, the steps needed to get an application running are 99.9999...% (you get the idea ☺) similar for both devices, so you should have no problem adapting the steps to suit the PrSoC extension board.

The tutorial is quite long, but you don’t need to read all of it, as it includes material that we cover in future labs. The chapters you should read are the following:

- Chapter 7: Cyclone V Overview
  - 7.2: Features of the HPS
  - 7.4: HPS-FPGA Interfaces
  - 7.5: HPS Address Map
  - 7.6: HPS Booting and FPGA Configuration
- Chapter 8: Using the Cyclone V – General Information
- Chapter 9: Using the Cyclone V – Hardware
  - 9.3: System Design with Qsys – HPS
  - 9.4: Generating the Qsys System
  - 9.5: Instantiating the Qsys System
  - 9.6: HPS DDR3 Pin Assignments
  - 9.7: Wiring the DE0-Nano-SoC
  - 9.8: Programming the FPGA
- Chapter 11: Using the Cyclone V – HPS – ARM – General
  - 11.2: Generating a Header File for HPS Peripherals
  - 11.3: HPS Programming Theory
- Chapter 12: Using the Cyclone V – HPS – ARM – Bare-metal

If you understand how the system is built and how the different components interact together, you will see that there is not much code to write for this lab. All the information you need can be found in the tutorial, but whenever in doubt, don’t hesitate to ask questions!

Your Task

Hardware

The goal is to create a minimal HPS system that can take a thermal image with the Lepton controller you implemented in lab 2.1. The system you have to implement is shown in Figure 1.

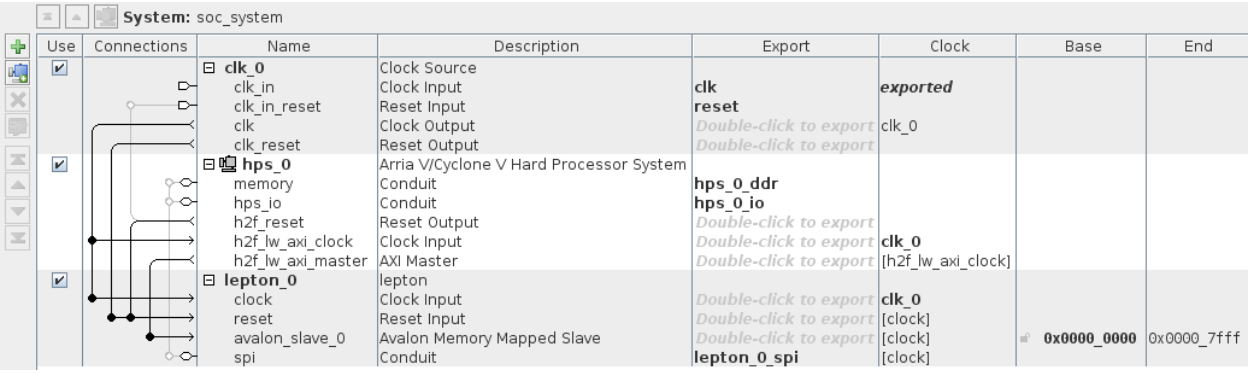

FIGURE 1. HYBRID QSYS SYSTEM TO IMPLEMENT

Your system must also be able to use the push button (`HPS_KEY_N`) connected to the HPS to toggle the LED (`HPS_LED`) connected to the HPS. This feature is on the house as the tutorial shows you how to configure the hardware to achieve this. Therefore, following the tutorial step-by-step should automatically make this feature work without issue.

To spare you the burden of entering the HPS' DDR3 timings by hand, we provide you a template where the "`SDRAM`" tab of the "`Arria V/Cyclone V Hard Processor System`" in Qsys is already filled in. Note though that you still need to configure all the other tabs!

Software

All the modifications needed for this lab can be implemented in the "`app.c`" and in "`lepton.c`" source files.

Lepton

Write the software needed to capture a thermal image with the Lepton. Unlike in lab 2.0 where we wrote the captured image to a *file*, we don't have access to any host filesystem when running bare-metal code on the HPS.

In order to see the output of the Lepton, you must modify the function that saves the data to a file to instead *print* the output on the serial console. All data outputted on the serial console can then be saved on your host machine by *manually* copy-pasting the output from the serial console into a file.

The ASCII nature of the PGM format we use to represent the image ensures that this simple copy-pasting procedure is adequate.

HPS_KEY_N & HPS_LED

Write the software needed to toggle the LED (`HPS_LED`) connected to the HPS whenever the push button (`HPS_KEY_N`) connected to the HPS is pressed. Again, this feature is on the house as the tutorial shows you how to do this, so if you follow all the steps, you should get this functionality for "free" ☺.