

Lecture 7 - Part 3

Getting up to speed with DE1-SoC board: HPS+FPGA Projects

Cristinel Ababei

Dept. of Electrical and Computer Engineering, Marquette University

1. Objective

The objective of this hands-on lecture-tutorial is to continue to learn about how to use the DE1-SoC board to create projects that are HPS+FPGA systems. This will be done with some more examples.

2. Prerequisites

I assume you have successfully done the examples from Parts 1,2 of this series of hands-on tutorials. The examples are numbered in continuation of the numbering from the previous part.

3. Examples

EXAMPLE #5: Control 7-Segment Display from HPS on DE1-SoC

The entire Quartus project folder of this example is **hps_fpga_7segment_vhdl_ex1.zip**, provided on the website of this course.

***Summary of the example:** This is a system that controls from the HPS side (running Linux off the microSD card) the 7-segment LED displays from the FPGA side of the FPGA SoC device on the DE1-SoC board. The functionality is a simple HEX counter that is also dimmed to a desired level. See Figure 1 below for more details.*

This example is adapted from a very nice example developed by Joel Bodenmann [1]. The example is not online anymore. A “snapshot” of the description that used to be online is included in the files provided with this lecture, as “**example5_Control 7-Segment Display from HPS on DE1-SoC _ Embedded.pdf**”.

FPGA HW:

This example **shows how to create your own Component in Qsys**. The Component, connected to the AMBA-AXI bus, is a controller for the 7-segment display, which will be controlled out of the Linux userspace code. Then, the example **shows how to implement the actual logic in VHDL of the new Component**. We have to listen to the Avalon bus and control the LEDs of the 7-segment display. Hence, second aspect of this new Component is to actually describe its behavior in VHDL (its creation in Qsys was more like its “declaration” only). Please see the VHDL source files in the archive provided.

HPS SW:

Finally, the example **shows how to interface from Linux userspace**. See **hps_fpga_7segment_vhdl_ex1\hps-c** in the provided archive with files for this lecture.

Once the FPGA is configured with the 7-segment controller we are able to talk to this peripheral from our user space. We use the Linux program **mmap** to map our newly created AMBA-AXI bus member to a virtual memory address. **See main.c for details.**

But, first we have to run a script that reads the QSys file and creates a C-Header containing information about the address space. The script comes with the board examples from Terasic and is called

generate_hps_qsys_header.sh. A copy of it is in: **hps_fpga_7segment_vhdl_ex1\hps-c**. Note that you have to run this script in the shell that came with the Quartus-II installation that can be open by executing **Embedded_Command_Shell.bat** in the installation directory. Launch the shell, then navigate to **hps_fpga_7segment_vhdl_ex1\hps-c** and type:

```
> ./generate_hps_qsys_header.sh
```

This creates **hps_0.h** file. The generated header file contains the following definitions among others:

```
#define OUTPUT_SEVEN_SEGMENT_COMPONENT_TYPE    seven_segment
#define OUTPUT_SEVEN_SEGMENT_COMPONENT_NAME    output_seven_segment
#define OUTPUT_SEVEN_SEGMENT_BASE             0x40000
#define OUTPUT_SEVEN_SEGMENT_SPAN             1024
#define OUTPUT_SEVEN_SEGMENT_END              0x403ff
```

We can now use these macros in our code to map the 7-segment controller to the virtual memory space. Again, **read the `main.c` to see details**. Then, compile using make. The resulting binary can be transferred to the booted Linux running on the HPS using SCP. **On execution it should dim the display to a modest value and start counting!**

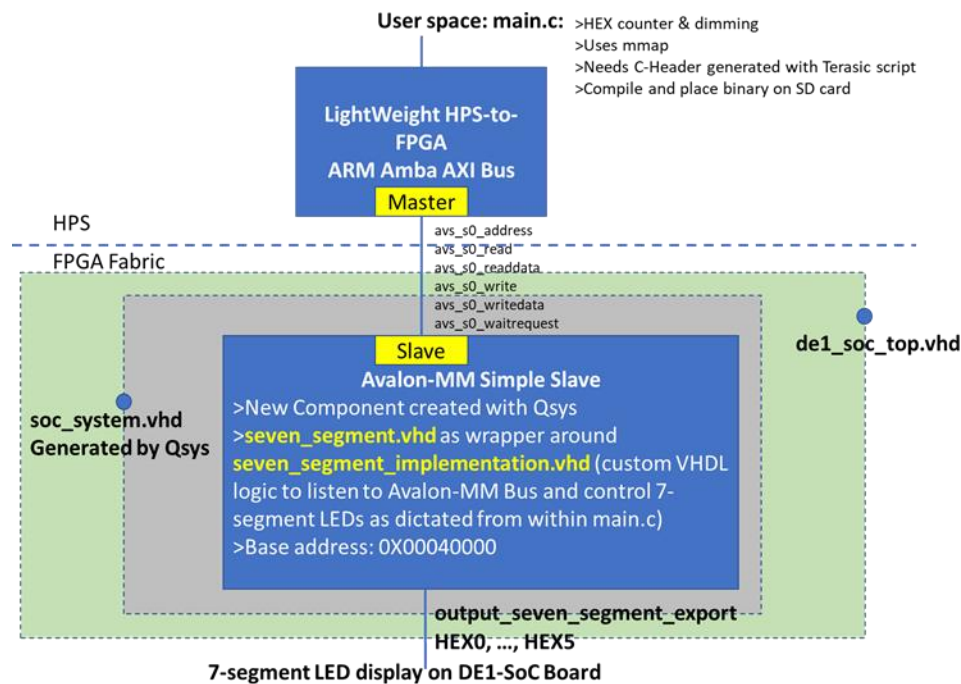


Figure 1: Visual representation to aid in explaining Example 5.

NOTES:

- A nice thing about this example is that it is all done in VHDL. See **de1_soc_top.vhd** for the top level design entity that is part of the FPGA portion of this example. This VHDL file is Joel's porting from Verilog to VHDL of the example from Terasic's CD. The VHDL version is clean and easy to understand.
- Good to know – Joel's Tip: *"I found it extremely useful to not implement the actual logic of a QSys component in the QSys created VHDL file but instead to create a custom VHDL file (I always gave them a `_implementation.vhd` suffix) and simply create an instance of that block in the QSys generated file. This way you don't have to re-generate the QSys code when you changed the implementation of the actual block. This turned out to be a huge time saver during development."*

EXAMPLE #6: Control of LEDR9 to make it blink

The entire Quartus project folder of this example is **hps_fpga_ghrd_vhdl_ex1.zip**, provided on the website of this course.

Summary of the example: This is a system that simply blinks LEDR9. It is more like a “place holder or template” project for DE1-SoC board that one can take and modify to add functionality to it.

This is adapted from a simple and nice example by Michael Fischer [2]. A “snapshot” of the description is included in the files provided with this lecture, as “**example6_emb4fun.pdf**” for your convenience.

This is a “bare-metal” example. It simply has a top level entity that instantiates two components: **heartbeat** and **soc_system**. The heartbeat is a standalone thing that simply drives the LEDR9 to make it blink. The soc_system does not do anything here; it’s here for us to use if we wanted. The nice thing is that it’s all ported to VHDL also. It’s inspired by the **Golden Hardware Reference Design (GHRD)** from RocketBoards, which Michael created for the Altera DE1-SoC Board. It is based on the Terasic Verilog example with some modifications and ported to VHDL.

So, in this current format this example has only **hw/** folder which contains the Quartus project with Qsys system. This is the counterpart of the usual **fpga-rtl/** folder in the examples from Terasic. We can use this example to build on top of it and create more complex projects. In those cases, we’ll need to make changes to it and to also add the **hps-c/** folder and write **main.c** which will be compiled and executed on the HPS part.

4. Summary

After doing the above examples, you should know more about HPS+FPGA systems.

5. References

- [1] Joel Bodenmann, Control 7-Segment Display from HPS on DE1-SoC, <https://silizium.io/7-segment/>
- [2] Michael Fischer, Altera DE1-SoC GHRD, <http://www.emb4fun.de/fpga/de1socghrd/index.html>