ECE 695R: System-On-Chip Design

Fall 2021

Abhishek Nayak
PUID: 33791158

Homework #2:

i) Describe (in 1-2 sentences) what happens when you 'assign base addresses' in Qsys?

Ans) The Qsys tool includes an address map tab that details the address range that each connected memory-mapped master uses to address each slave component to which it interfaces. When we use 'assign base addresses', the tool automatically assigns unique base addresses to all the slave interfaces so that the conflicts due to overlapping of addresses are avoided.

ii) What is the relationship between the module 'demo' described in the file demo.v and the top level component 'nios_system' in the RTL code generated by Qsys?

Ans) 'nios_system' is an RTL code for the top-level Qsys system (in this case - the SoC with NIOS II processor) that instantiates each submodule in the system.
The module 'demo' is the top block in demo.v file which contains the SoC (ie 'nios_system') and the PLLs; also connects the SoC to the pins of the FPGA on the FPGA board. As we can see in the demo.v file, the 'nios_system' is instantiated.

iii) Go through the file nios_system.sopcinfo that is automatically generated by Qsys. Briefly (in 3-4 sentences) describe the contents of this file.

Ans) The nios_system.sopcinfo file describes the components and connections in our NIOS II system. It also describes the parameterization of each component in the NIOS II system.
For example - the module 'nios2_qsys_0', different module parameters are listed, such as BIG_ENDIAN = 0 (off), CPU_FREQ_50000000

Essentially, the sopcinfo file contains system configuration information. Parsing its contents will enable us to get requirements when developing software drivers for Qsys components.

iv) Locate the file system.h in the software/demo_board_bsp directory. The program that you wrote uses hardwired addresses (eg- 0x01001020) to access the LEDs and Switches on the DE2 board. Can you eliminate the use of these hardwired addresses by taking advantage of the macros in system.h? Describe how.

Ans) Yes, we can eliminate the use of hardwired addresses.

The macros RLEDS_BASE and SWITCHES_BASE are defined in system.h. Now, we replace the Red LEDs & Switches address in the code with the above macros/aliases defined in system.h code.
The system.h header file also needs to be sourced in the code.

The code now →

```
#include "system.h"
int main()
{
    int *LEDS = (int*) RLEDS_BASE;
    volatile int *Switches = (int*) SWITCHES_BASE;
    .
    .
    .
```

v) Locate the header file for the Pixel Buffer device driver. (altera_up_avalon_pixel_buffer_dma.h). Describe (in 2-3 sentences) the significance of this file and its contents.

Ans) The header file specifically consists of the device structures and the instantiation of the functions which are used to display in the VGA monitor.
For example - The function 'alt_up_pixel_buffer_dma_draw_hline' draws a vertical line of a given color between points (x, y0) and (x, y1).

The functions are declared in the header file but are defined in the .c file (for ex- alt_up_pixel_buffer_dma_draw_hline.c).
Also, the driver instances use the structures to hold its associated states.

vi) In the interactive lecture, we used the 'Nios IIe' version of the processor. Upgrade the processor to 'Nios IIf' and re-run the program. Does the program still work? If not, why not? How can you fix it?

Ans) The program did not work after upgrading the processor to 'Nios IIf'.

Reason -
There is a cache memory in 'Nios IIf', which is absent in 'Nios IIe'
The data cache cannot correctly access the memory-mapped I/O since the Red LEDs peripherals are assigned the address 0x10201020.

To fix the error -
Cache bypassing is required.
For the Nios IIf system, the MSB bit is reserved for cache bypass.
If MSB = 1 → Cache is bypassed.
If MSB = 0 → cache is not bypassed (The error with 0x10201020)

If the MSB is set to 1, the program executes successfully.
Hence the new address for Red LEDs should be updated to 0x90201020.