MP-0

# nes\_bootloader.c

No idea what the structure looks like. Need to draw a diagram.

NESCore\_Callback\_OutputFrame() gets called when we are drawing a row that isn’t the first row and isn’t a vertical sync.

# System Assembly View

The Green Boxes

1. 32b GP AXI Slave Ports

We can enable GP0 and GP1 interfaces. We can modify the address ranges of each. Controlling memory is always useful in an embedded system. It seems like we can set memory addresses ourselves or we can let the program decide them for us.

1. I/O Peripherals

We can enable or disable peripherals. This is useful because it lets us optimize our designs and be flexible to modifications.

1. General Settings

It has a lot of options that don’t fit anywhere else. We can change the baud rate of the UART as well as Enabling power on reset. These are very nice features to extend the ability of your design.

# PS and PL

We think the buttons, LEDs, and switches are connected via the PL subsystem. We wouldn’t write c code to change how the bus works so this doesn’t fall into the PS subsystem. Since it isn’t PS, it must be PL. Also, we know how the bus works so we make our design implement the bus protocol. We don’t implement the bus protocol in code.

The AXI\_GPIO IP core acts as an interface. The bus doesn’t care what is on the other side of the interface. There is a standard when talking to AXI\_GPIO IP cores. Since buttons, switches, and LEDs are all GPIO on a bus following the AXI protocol, they all get labeled as AXI\_GPIO.

# Read the Switches

In order to read the switches, we have to start off with the base address of the switches. In the Addresses tab we can see that “SWs\_8Bits” has a Base Name of C\_BASEADDR and a value of 0x4120FFFF with a size of 64k. Now taking that information over to the axi\_gpio datasheet, on page 8 it describes the registers of the IP. It describes 4 registers. 2 of which are data registers (one for each channel, if enabled) and 2 for tristate values. Reading or writing is done with the data registers. To read and write you use the following equation. C\_BASEADDR + 0x00. We know that the switches C\_BASEADDR is 0x4120FFFF. Therefore, 0x4120FFFF + 0x00 = 0x4120FFFF. Now simply read the data at that address.

# Print()

The print() function is a watered down version on printf(). Print() copies a string byte for byte while printf() does a lot more, such as format specifiers and data type conversions.

# LED’s Hello World

Cyclone check:

We found what fraction out of 255 each color channel was and then applied that percent to the 16 bits that we have to work with.