數位電路實驗 Lab-3 教學手冊 – 錄音機

Team #2 B02901027 茅耀文, B02901178 江誠敏, B02901179 黃凱祺

December 3, 2015

0 前言

這次的實驗我們主要是使用 Nios 實做的，因此主要解說 Nios 的細節。

1 Nios

先說使用 Nios 的優點。

1. 可以用 C/C++ 寫而非相對難寫很多的 Verilog。
2. Debug 較容易，可以 `printf` 印出數值等等。
3. Verilog Compile 一次大概要 3 min, C/C++ 只需幾秒。
4. 程序邏輯相對簡單很多。

但當然 Nios 還是有很致命的缺點的。

1. Nios 非常慢，慢到讓你想哭的那種慢。如果有 Real time 的需求 (如 Audio, Video) 最好考慮一下 (雖然我們兩個都用了)。
2. 系統有點不穩，至今原因不明 (有可能是我們寫錯了…)。
3. 要和 Verilog 寫的元件接比較不容易。

最後一點好在 Qsys 裡有提供不少的元件可以使用。
記得要升級到 Quartus 15.1 版，15.0 的 Qsys 似乎有點 Bug 。

1.1 Nios 開發流程

大致上如下：

1. 用 Qsys 合出 Nios 以及周邊的元件。

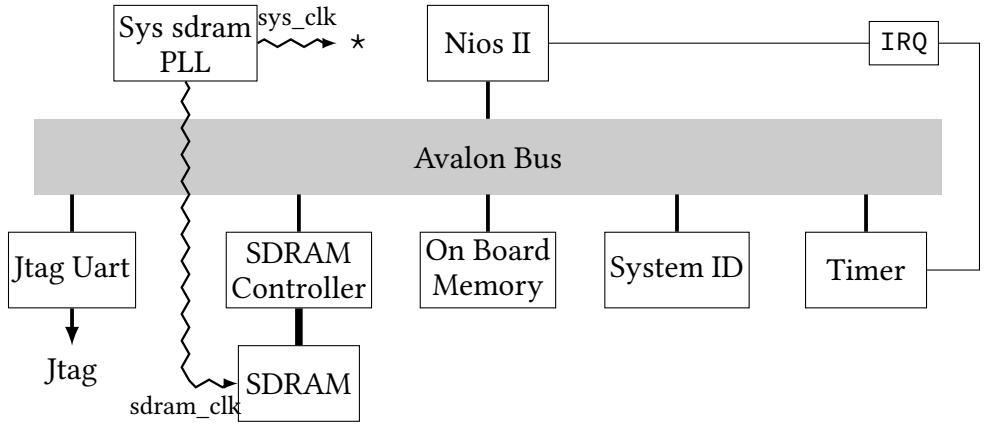
2. 在 Top Module instantiate Qsys 合出來的電路，並將 Signal 接起來。
3. Compile Verilog 並燒上 DE2-115。
4. 編寫 C Code。
5. Compile C code 並燒上 Nios。

以下會簡述一下這些過程。可以參考一下網站上的 ppt[1]。

1.2 Qsys

基本上這個步驟不難，就看你要哪些元件，把他加上去並將線接好即可，以下分各個元件來講。

1.2.1 Nios



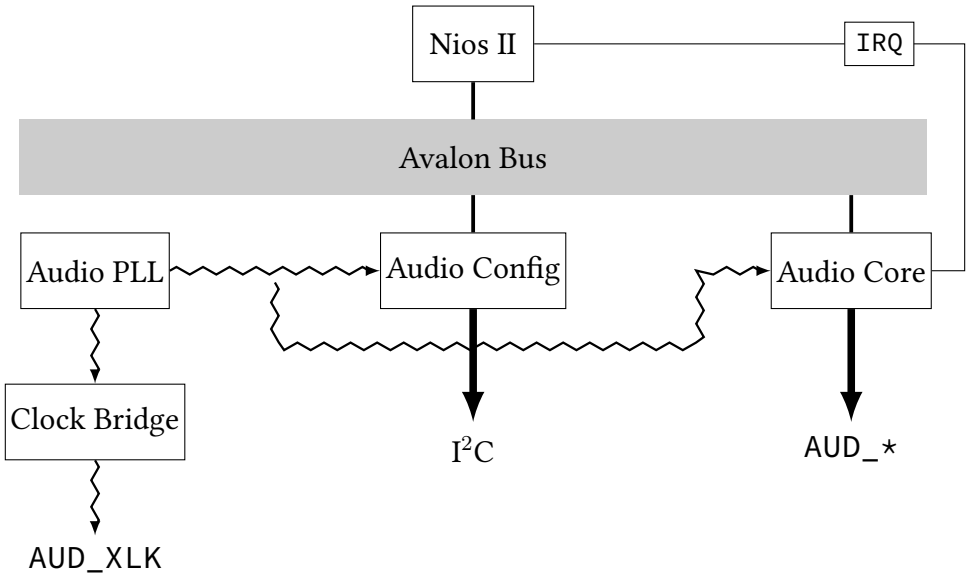
1. Nios: 就是 CPU 啦，另外 `clk` 最好高一點 (聽說可以超到 100 MHz，我們使用 75 MHz)，不然 Nios 本身真的太慢了。
2. Jtag Uart: 這個是用來燒 C Code 進 Nios 還有 Debug 的必要元件。
3. SDRAM, On Board Memory: 提供 Nios 記憶體。DE2-115 上 On Board Memory 大概可用 200 KiB, SRAM 有 2 MiB, SDRAM 則有 128 MiB。建議是使用 SDRAM，記得要把 Nios 的 Vector 設在 SDRAM 上。
4. Sys SDRAM PLL: 如果要用 SDRAM, **SDRAM 的 Clock**[2] **必須比其他人快** 3 ns，所以要用此 PLL 合出需要的 Clock ，之後**一般的元件都會接在 sys_clk 上**，記得 SDRAM (**非 SDRAM Controller!**) 的 clk 要是 sdram_clk 。

[1] *My First Nios II for Altera DE2-115 Board.* URL: https://www.google.com.tw/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwjB_JPNsb3JAhWKo5QKHWWABDcQFggcMAA&url=http://dclab.ee.ntu.edu.tw/static/Document/Project/Project_3.pptx&usg=AFQjCNFP3SZgT7ZRoMYs7dwXIalbknC76Q&sig2=7vCY-3MqKkWWwfhvJMjM6w.

[2] *Using the SDRAM on Altera's DE2-115 Board.* URL: ftp://ftp.altera.com/up/pub/Altera_Material/9.0/Tutorials/Verilog/DE2-115/Using_the_SDRAM.pdf.

5. System ID Peripherals: 一個產生 TimeStamp 的元件，當你 Qsys 更新時會使 C compiler 知道有改動。
6. Timer: 可供 Nios 計時用。

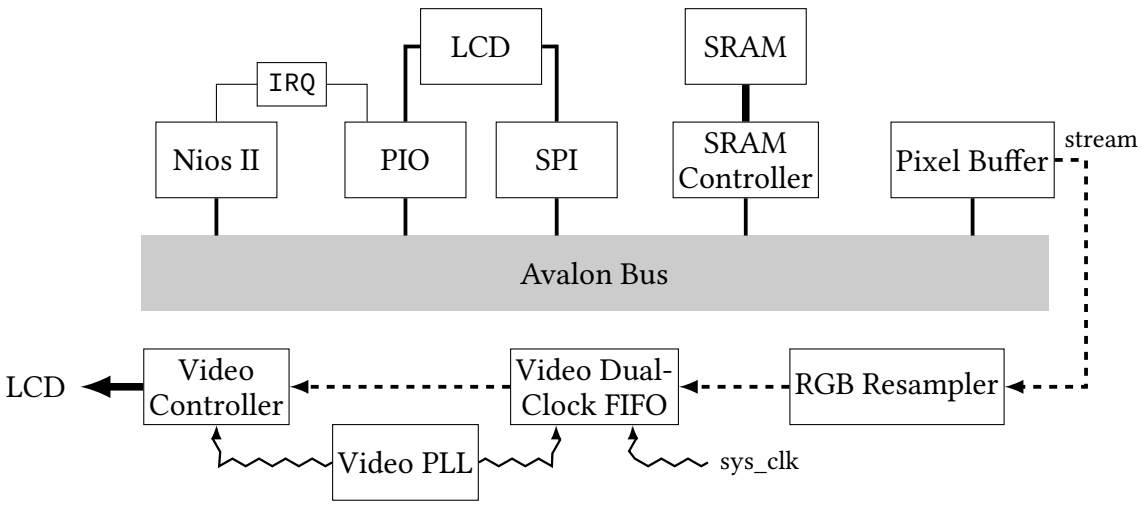
1.2.2 Audio



1. Audio PLL: Qsys 裡內建幫你生 Audio Clock 的元件。
2. Audio Config: 全名 Audio and Video Config，用來幫你初始 Audio Chip 的元件。可以勾選 Auto Initialize ，這樣 Nios 可以完全不管他。
3. Audio Core: 控置 Audio 的核心。之後 Nios 讀/寫 Audio 都透過他。

這邊我們一樣會讓 WM8731 (Audio Chip) 在 Master 模式運作。記得 export 出去的 signal (I^2C, AUD_*) 要接對。

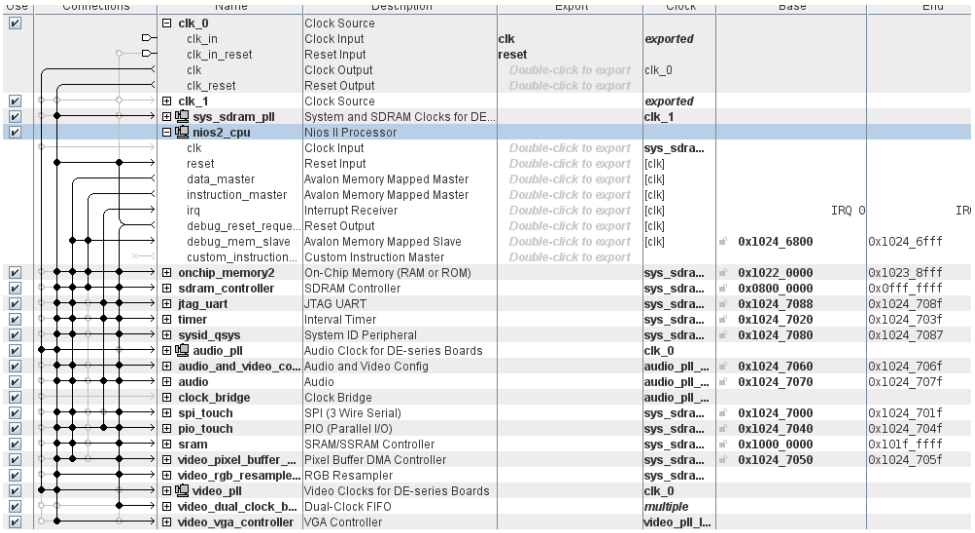
1.2.3 Video



1. SRAM Controller: SRAM 控置器，SRAM 有 2 MiB，大小恰恰好可以當一個兩個 buffer (front/back) ，480×800 的 pixel buffer。
2. Pixel Buffer: 之後 Nios 就直接透過他來畫圖。記得 pixel buffer address 要設在 sram 的位置。
3. RGB Resampler: 把 SRAM 16 bit RGB(565) 轉成 Video Controller 接受的格式。
4. Video Dual Clock FIFO: 因為 Video Controller 和 Pixel Buffer 的 Clock 不一樣，必須接個 FIFO 。
5. Video Controller: 幫你輸出訊號給 LCD, VGA 等等。
6. PIO, SPI: 偵測 LCD 觸控板的元件。

這邊比較複雜，建議可以直接接個 Video test pattern generator 在 Video Controller 上看 LCD 會不會有彩虹圖案，來確定沒有接錯。

將全部接起來後大概就是這樣：



成功後還滿有成就感的。

1.3 Top Level

之後請記得調整 Top Level，加入 Qsys 產出的檔案。

```
module DE2_115(
        input CLOCK_50,
    ...
        output SRAM_OE_N,
        output SRAM_UB_N,
        output SRAM_WE_N,
);
```

```
8
9  Core (
10     .audio_and_video_config_external_interface_SDAT(I2C_SDAT),
11     .audio_and_video_config_external_interface_SCLK(I2C_SCLK),
12     .audio_external_interface_ADCLRCK(AUD_ADCLRCK),
13     .audio_external_interface_ADCDAT(AUD_ADCDAT),
14     ...
15     .spi_touch_external_SS_n(adc_sel_n),
16     .pio_touch_external_connection_export(GPIO[0])
17 );
18 ...
19 endmodule
```

2 Programming in C

2.1 Start Project

基本上就按照 ppt[3] 的做法就可以了。會有兩個 project 產生，你的 project 還有他對應的 bsp project。之後如果有更新 Qsys 記得要

```
1 nios2-bsp-generate-files --settings=<bsp dir>/settings.bsp \
2   --bsp-dir=<bsp dir>
```

2.2 Drivers

Driver 的使用說明基本上上網查就有了，比如說 Audio Core[4] 和 Pixel Buffer[5] 。

或者也可以直接看 BSP Project 裡的 `.h` 檔。

2.3 `io.h`

基本上所有和 Avalon Bus 上的元件的溝通都可以透過 Memory Map 的方式達成，而 `io.h` 裡面就定義了兩個 marco `IORD(addr, reg)`, `IOWR(addr, reg, val)` 可以使用。

但當然，如果以經有現成的 Driver 可用就直接用 Driver 裡的函式即可。

[3] *My First Nios II for Altera DE2-115 Board.*

[4] *Audio core for Altera DE-Series Boards.* URL: ftp://ftp.altera.com/up/pub/Altera_Material/15.0/University_Program_IP_Cores/Audio_Video/Audio.pdf.

[5] *Pixel Buffer for Altera DE2/DE1 Boards.* URL: ftp://ftp.altera.com/up/pub/Altera_Material/8.0/University_Program_IP_Cores/Pixel_Buffer.pdf.

2.4 `system.h`

這裡面定義了元件的位置、IRQ ID 等等的 marco 。

2.5 `sys/alt_irq.h`

當某個 Signal 被觸發時，可以透過送 IRQ 給 CPU，讓 CPU 直接跳去執行某一個函式，達到 Real time 的效果。

可以參考官方文件[6]的說明。

基本上 Audio 最好使用 IRQ ，否則很容易有爆音的現象。要注意 Interrupt 不能執行過長。

2.6 μC-OS II

μC-OS II 是某個被附在 Eclipse project template 的 real time OS。相關的說明可以參考說明文件[7] ，雖然感覺功能不太強，但是免強還可以用。

2.7 LCD Touch Screen

我們是用 SPI (3 Wires) 和 LCD 的 ADC 溝通，可以參考 LCD[8] 的文件以及 Altera SPI[9] 的說明。

另外可以再用 PIO 做一個 IRQ ，便可以偵測螢幕被觸控的時間。

2.8 Real time

基本上 Nios 的速度真的太慢了，能保持 Audio Real time 已經幾乎是極限了。

Video 的話大概要 30 Hz以上才能有動畫效果，10 Hz以上觸控事件才可以被準確判定。最後我們 LCD 的更新頻率大概只有 3 Hz左右。不確定透過一些優化後可以達到多少。

[6] *Exception Handling.* URL: https://www.altera.com/ja_JP/pdfs/literature/hb/nios2/n2sw_nii52006.pdf?#page=7.

[7] *μC/OS-II Documentation.* URL: https://doc.micrium.com/display/osiidoc/home.

[8] *4.3" LCD Touch Panel Package.* URL: http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=213&FID=b226168825c32dd5d7064e9a57f42b0b.

[9] *Embedded Peripherals IP User Guide.* URL: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_embedded_ip.pdf?#page=121.

3 過程中遇到的問題

下面列了一些我們在過程中遇到的問題。

3.1 Verilog

- `.sdc` 檔 (TimeQuest Timing Analyzer) 一定要記得加，沒加有一定機率產生其他奇怪的問題。以下是一個範例 (就是助教 lab1 給的)。

```
create_clock -period 20 [get_ports CLOCK_50]
create_clock -period 20 [get_ports CLOCK2_50]
create_clock -period 20 [get_ports CLOCK3_50]
derive_pll_clocks
derive_clock_uncertainty
```

3.2 Audio

- WM8731 DACDAT 在送完 16 個 Bit 後一定要設為 0，否則出來都是雜音。官方說明文件都沒有寫。
- WM8731 DACDAT 的第一個 bit 一定要取反，換句話說 `dacdat = adcdat ^ (1 << 15)`。我們完全不知道為什麼會有這種問題，雖然好像別組都沒有這個問題，但我們用 Qsys 的 Audio Core 也會有這個問題 (除非 Quartus 的 Audio Core 和我們都寫錯了)。
- 內差的時後 Nios 太慢導致爆音。這只能靠優化了，我們的做法是盡量用位元運算取代除法 (`/32 = >>5` 等等)。

3.3 Video

- Qsys 產出的 Video Core VGA Signal VGA_BLANK 不知為啥似乎被取反了。
- LCD 沒有反應時可以換一條 IDE 排線或是動一動 IDE 排線看看，實驗室裡的 IDE 排線很多都接觸不良。
- Terasic LTM 我們用 Qsys 裡的 Video Controller 完全跑不動，或許是 Video Controller 寫壞了。但 LTM 基本上把硬體資訊鎖起來了，連腳位都很難查，Debug 不能。
- LCD 色彩不對，在 $\mathrm{rgb}(128, 128, 128)$ 基本上已經是全黑的了，這個需要自己做 Gamma Correction 或是用 Qsys 裡的套件。