逢 甲 大 學

資訊工程學系專題報告

MP3 Player on Nios

學　　生：　李 丕 耀（四乙）

指 導 教 授　：　　王 益 文

中華民國九十二年十二月

目　　錄

圖表目錄

摘要

使用altera nios development board去做一個mp3 player，使用Nios soft core cpu去decode出pcm raw data，然後透過dac轉成analog 訊號給喇叭或耳機輸出。

第一章 導論

1.1 MP3

MP3是現在現在很普及使用的音樂壓縮格式，從網路上傳輸到一般使用者做音樂壓縮儲存等應用，相關的周邊商品也越來越普及、價格也越來越低廉，雖然現在有眾多知名的音樂壓縮格式，但是現在最常見、通用家電產品音樂壓縮格式支援的就是mp3，從家用DVD/VCD player到隨身聽，甚至隨身碟也附上mp3 player的功能。因為mp3的高壓縮比，所以可以用較小容量的儲存媒體，像是應用flash memory的產品，除了低耗電，還有不怕震動的好處；當然也有其他如掛上硬碟、光碟機的應用，通常比較偏向家中電器的應用，且可以容納驚人數量的音樂也是吸引人的地方。

MP3在壓縮上有壓縮比的差別，常見的如48kbps、64kbps、96kbps、128kbps…320kbps，另外有固定幾種頻率，單、雙聲道可供選擇，但是前面提到的壓縮比決定了每秒的音樂要用多大的資料量來存。

1.2 MP3格式

· Layer Ⅰ、Layer Ⅱ、Layer Ⅲ

· Mono、Dual Mono、Stereo、Joint Stereo

· MPEG1

(32kHz, 44.1kHz, 48kHz):

32 40 48 56 64 80 96 112 128

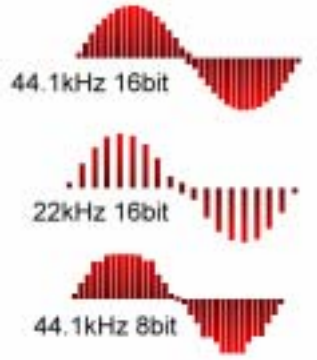
160 192 224 256 320

MP3 Bitrate	效果比較
8 / 16 / 24	電話
32 / 48	AM 收音機
56 / 64 / 80	FM 收音機
96 / 112 / 128	接近 CD
160 / 192 / 224 / 256/ 320	更接近CD

· MPEG2

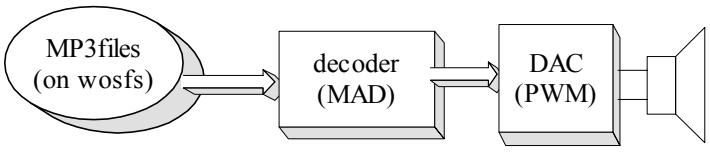
(16kHz, 22.05kHz, 24kHz):

8 16 24 32 ~160



1.3 MP3 Player on Nios

有機會接觸到Altera NIOS這顆soft-core cpu和sopc builder這種FPGA的環境，所以決定嘗試在這平台上實作mp3 player。decoder我採用open source的MAD(Mpeg Audio Decoder)project



圖表 1 簡易架構圖

第二章 使用平台

開發環境為

Altera Nios Development Kit

Nios 3.0 32bit cpu & SDK

開發過程中使用到

APEX™ 20KE device (EP20K200EFC484)

dual seven segmenta

serial

JTAG

6 push bottom

3 pin pio(for analog out)

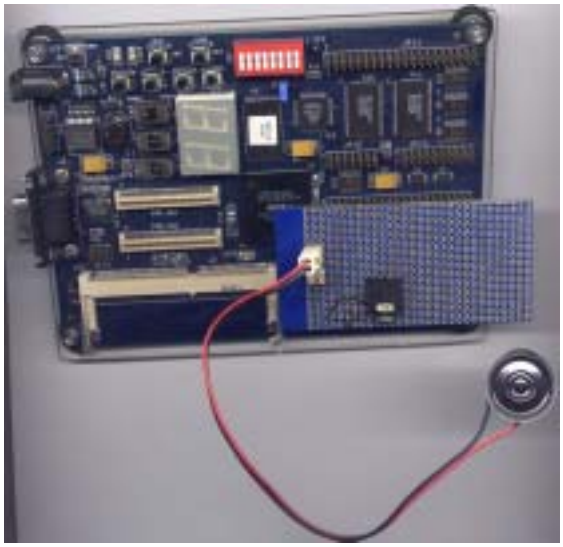
flash memory 1MB

sram 256KB等

後來有加入Stratix development board

Stratix EP1S10F780C6

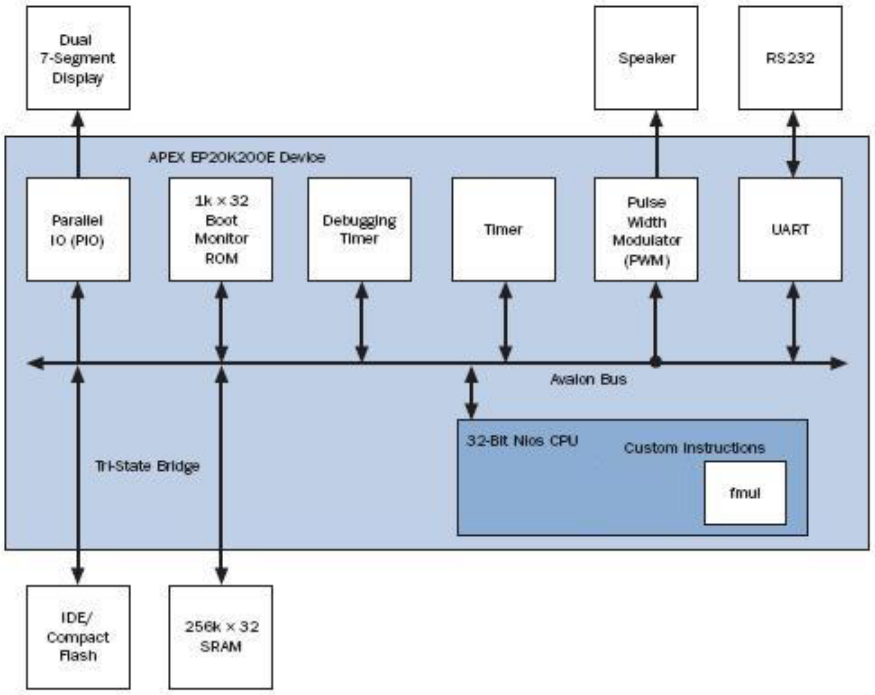
記憶體容量較大、速度也較快



圖表 2 成品照片

圖表 3 stratix 版成品照片

第三章 系統架構



圖表 4 mp3 player 架構圖

在實作這個mp3 player時，選擇MAD這個軟體decoder，首先就是搭配nios sdk porting給nios cpu使用，然後做fmul custom instruction來提升效能，然後MAD解碼mp3資料輸出PCM data給PWM模組轉成analog方波訊號，這樣就可以接上喇叭發聲了。

第四章 軟體部份

4.1 MAD—Mpeg Audio Decoder

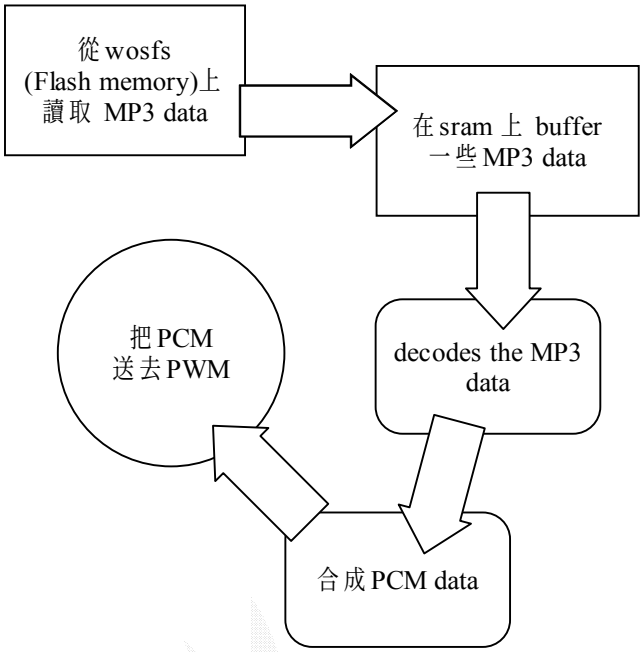
通常市面上常見的MP3 player的processor通常是用來控制播放和移動資料，然後搭配一個MP3 decoder的硬體電路去解碼，在這裡則是全部由Nios這顆cpu來處理。

選擇使用MAD的好處在於：

- 100% fixed-point integer運算

  (對於embedded system上要實作較容易)
- 24-bit PCM output
- (在此只用16-bit，但以後可以延伸做成24-bit)
- 依照ISO/IEC standards
- GPL授權使用
- source主要在GNU C的環境下發展，且可以OS independent，適合使用在沒有OS的device上(MAD本身的範例是high level sdk 控制，另外有low level sdk比較適合沒有OS的系統來使用)

實作上執行的流程是:



圖表 5 decode 流程圖

最耗cpu運算的部份在於合成subbands，因此mad中的mad_synth_frame這個function需要想辦法加速，其中的f_mul和mad_f_mul兩個function用的很多。f_mul、mad_f_mul這兩個function是MAD用來以整數相乘模擬浮點數相乘用的，用了很多的位移、加法、乘法還有OR運算，這些很容易從hardware的方式去實作，所以做成custom instruction。因為兩個function很相似，所以做成一個custom instruction就可以了，在後面硬體的部份會再提到。

4.2 nios_audio.c

這個程式提供了處理audio stream的API，也處理關於中斷的問題。若遇到PCM data是stereo的，則將兩個聲道合併成一個。Play/Stop ,mute都是透過這裡控制的,sampling rate也是從這裡設定。

4.3 Nios_audio_asm.s

這是一個小ISR被timer利用來從output buffer 讀取資料到audio port。

Internal Operation:

1) gOutBuffHeadPtr is checked against gOutBuffTailPtr. If they are the same, then no action happens (goto step 5).

2) gOutBuffHeadPtr is checked against the (gOutBuffEndPtr + 2).If gOutBuffHeadPtr is at (gOutBuffEndPtr + 2), then gOutBuffHeadPtr is reset to gOutBuffBeginPtr (buffer beginning).

3) Value at gOutBufferHeadPtr is copied to location gOutputPort.

4) gOutBufferHeadPtr is incremented by 2 (sizeof hword).

5) Interrupt is cleared and ISR returns.

第五章 硬體部份

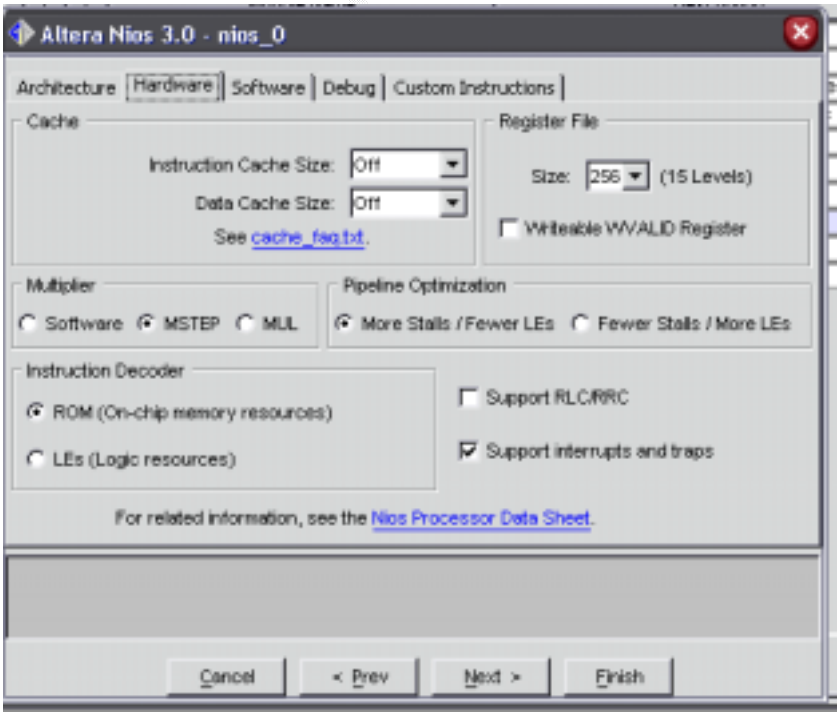
5.1 Nios soft-core CPU設定

使用Altera Nios 3.0 32-bit cpu:

32bit ALU,register,and data bus

32bit addressing(maximum)

standard features/Average LE usage



圖表 6 nios cpu設定

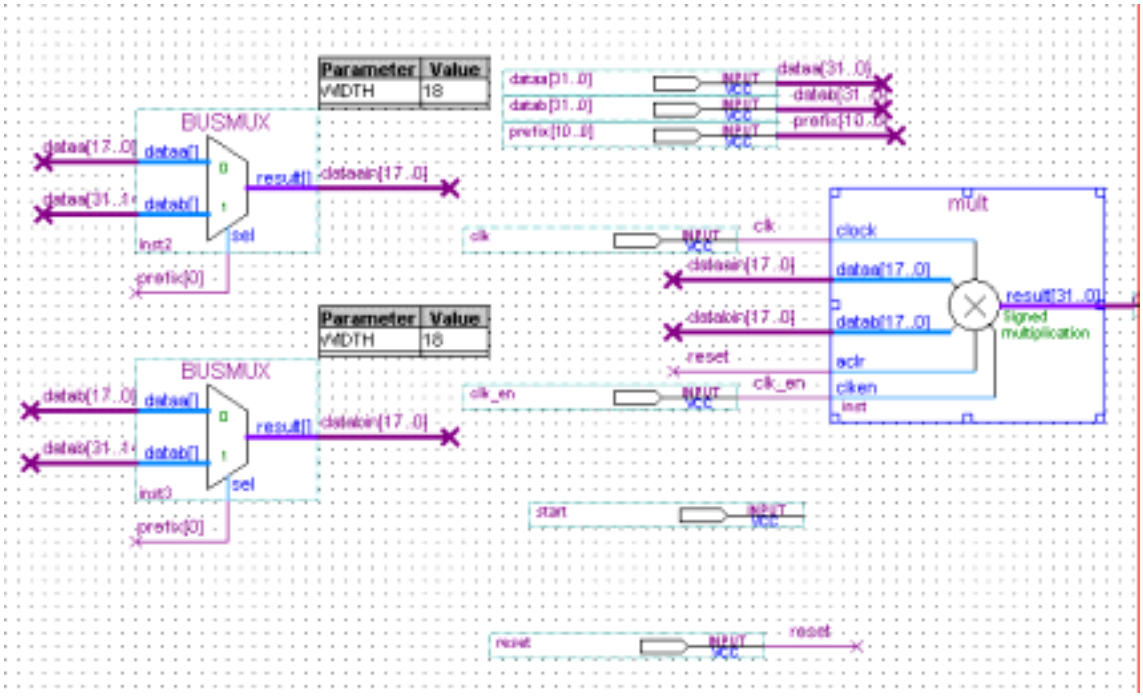
5.2 fmul custom instruction

原來的function

```
#define mad_f_mul(x, y)
((((x) + 0x00002000L)>> 14) × (((y) + 0x00002000L)>> 14))
#define f_mul(x,y)
(((x)| 0x0001FFFFL) × ((y)| 0x0001FFFFL))
```

做成如下圖的fmul custom instruction，使用bus mux來達到將兩個function整合成一個custom instruction。

```
#define f_mul(x,y) nm_fmul((x),(y))
#define mad_f_mul(x,y) nm_fmul_pfx(1,(x),(y))
```



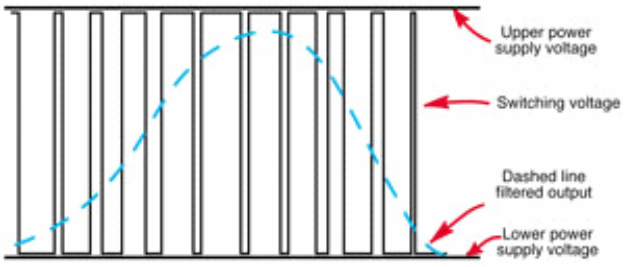
圖表 7 fmul custom instruction

5.3 PWM-Pulse Width Modulation

這裡使用PWM來解決DAC這個環節，pwm.v這部份就是接到東西，然後發出聲音，pwm模組內部有兩組 16-bit write-only registers 可供cpu使用：

period

pulse width



PWM module和cpu之間傳資料使用一條16 bit的data_from_cpu on Avalon bus。

每次輸出波形的cycle，PWM模組會產生periodic_irq中斷，nios可以用這個中斷去寫入下一個值進”pulse width” register。當完成寫入後，清除irq-signal。當period register為0時，pwm不發出聲音，也不會發出中斷訊號。

當wave_counter < pulse width時，output is high

5.4 增加sys_clk

原本apex是跑33.333Mhz的，因為解碼上速度還不是很夠，遇到64KBPS以上的就跟不上real time了，所以開始尋求讓sys_clk更快的方法。使用quartus_sh –dse叫出Altera Design space explorer，可以花些時間去讓他換用不同的seeds 去找出比較好的結果，也因此才找到目前這一個只有一個critical path且workable on 44.444Mhz的解，目前44MHz可能是這張板子目前的極限了。

後來有使用到 Stratix這張板子，可以跑到66Mhz，更快還沒有測試。

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	Source Name
1	Clock Setup: 'connector_pll:inst2|altpll:altpll|clock_out:clock0'	-0.191 ns	44.44 MHz (period = 22.500 ns)	44.27 MHz (period = 22.691 ns)	cpu32:inst|nios_cpu:c
2	Clock Setup: 'connector_pll:inst2|altpll:altpll|clock_out:clock0'	0.223 ns	133.33 MHz (period = 7.500 ns)	137.42 MHz (period = 7.277 ns)	cpu32:inst|audio_pwm:1
3	Worst-case tsu	N/A	None	15.639 ns	RST_D[1]
4	Worst-case tco	N/A	None	10.655 ns	cpu32:inst|nios_cpu_avalo
5	Worst-case th	N/A	None	-2.558 ns	FLD_CLEAR_N
6	Worst-case minimum tco	N/A	None	3.256 ns	cpu32:inst|nios_cpu_avalo

圖表 8 Timing Analyzer Summary

	Logic elements	Memory bits
Original	2,426/8,320	26,496/106,496
fmul custom instruction	3,153/8,320	26,496/106,496

圖表 9 gate count report

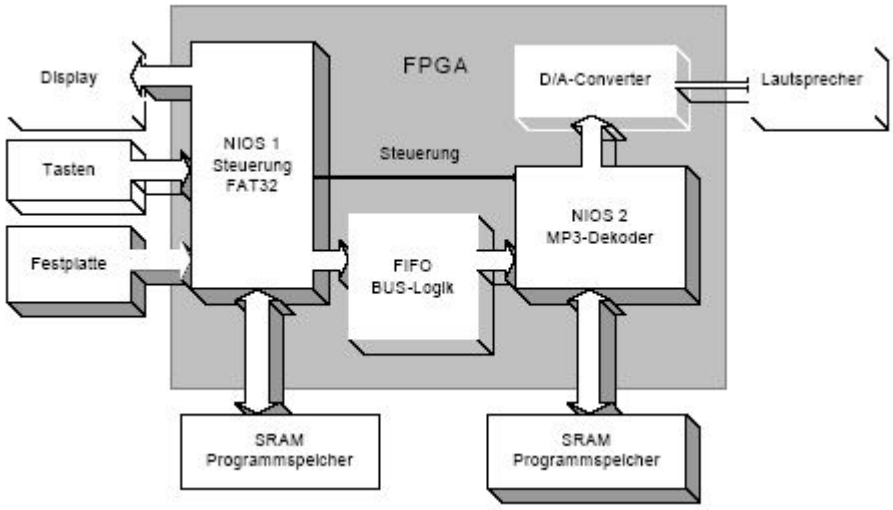
第六章 問題討論

原本Apex dev board是33.333Mhz，目前在Apex板子我最多能跑到44.444Mhz（有一個critical path）但還是可以正常使用，能順暢的播放96Kbps 單聲道的mp3檔案，超過就很容易聲音變得延遲不自然，還有雙聲道解碼上也是資料量有增加變得很吃緊，如果換到更powerful的板子上，也許可以支援更廣範圍的格式到320Kbps。

解單聲道沒有問題，但是在解Stereo的時候，往往聲音都變成慢速放音，本來以為是decode perfromance上出問題，跟不上real time。後來發現同樣mp3 bitrate下不同取樣頻率比較時，32Khz可以正常放stereo，但是44.1Khz就不行，這才想起來pwm那邊有個動作把兩個聲道的pcm data merge成一個，這個動作並沒有特別處理去加速，變成是拖慢pwm要放音的效率。

DAC用pwm來解是很方便的省掉了外部電路，不過現在只有單聲道，是否能負荷到雙聲道是個待解的問題；一方面可以先把pwm改成兩個各自解一個聲道，另外我比較傾向於希望把音響用16bit 雙聲道DAC掛上去，這方面要實作起來照一些業界常用的I^2C格式和一些spec應該有希望達成。

在讀取MP3檔案方面，目前是使用altera範例程式中用到的wosfs，這個簡單的file system是based on flash memory的，每次要放測試檔案進去都要先經過轉換成適當的格式，然後用nios-run程式來透過板子的serial console傳進去，一方面耗時、一方面板子上的flash memory扣除fpga硬體、軟體initial用掉一大部分，剩下的空間只有幾百KB。這方面要嘗試把更大的flash memory掛上來，可能要改寫一些sdk來支援，若是掛上ide界面，把硬碟或光碟弄上來讀取會更加方便，在蒐集資料時有看到德國的一份mp3 player實作介紹，就有成功把ide硬碟掛上去使用，不過他們的做法是再掛第二個nios cpu來處理Fat32。



圖表 10 德國 nios mp3 例子

第七章 結論

這個專題是在暑假開始執行,從收集資料到開始遭遇到很多難題，最後是使用altera的範例程式，才出現這個成品。這是一個成功可以播放mp3檔案的mp3 player，支援的壓縮比不是很完整，但不是decoder的問題,只是計算上不夠快到能夠順暢的real time播放。從這個專題中，對於altera nios這個平台比較熟悉一些了，也在遇到問題時跟Altera內部FAE連絡上，得到不少很有幫助的建議跟方向。在i/o處理跟timing上還有很多要學習去處理，像是sdram要掛起來正常的run程式就不是很容易。

計畫延伸：

A. 參考opencores裡的OCIDEC (OpenCores IDE Controller)把ide interface做起來，嘗試來使用cd-rom或是hard disk來當作mp3 儲存媒體，以後要做encoder也比較方便。

B. 另外掛一個較大容量的flash memory。

C. 考慮加上用音響用的雙聲道16-bit dac，利用pio輸出相容serial信號，目前已知可能要做I^2C controller，這個目前oopencores也有成品可以參考。

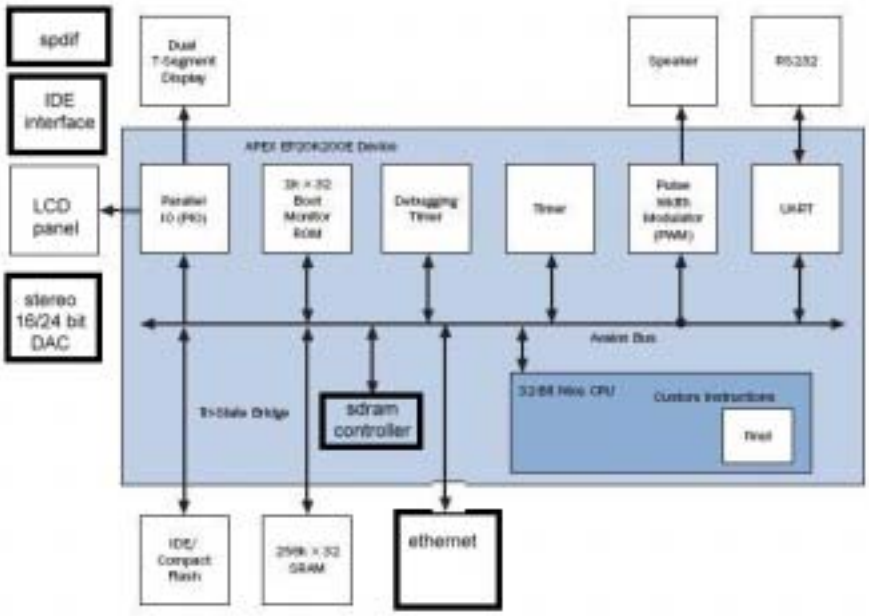
D. spdif界面輸出，可以外接一些標準dac音響裝置、音效卡。

E. 想要做到完整能解所有mp3壓縮比，除了增加sys clock(這在FPGA上似乎不容易達到需求)，還有一個方法就是將decoder硬體化，目前有xilinx放出來的mp3 example code可以參考。

F. 加上ethernet界面，寫socket程式，來變成mp3 streaming server/client雙用。

G. 假如只在soft-core用cpu來跑decoder，要支援其他AAC、WMA、OGG等格式是比較容易達成，比較保有彈性，但恐怕做出來都是playable，但是不是full support，較小壓縮比的檔案可能都不能real-time搞定。但是如果掛上較大記憶體如sdram，可能可以嘗試先解碼buffer到sdram上，不過apex的pll(可以倍頻、除頻用)只有兩個，目前已經不夠用，要掛sdram都無法(目前我sdram僅能跑在100MHz才會穩定，且只能存取資料，要跑程式在上面會失敗，timing還沒有完全抓對)。不過其他更高階的板子如stratix的pll都很充裕，所以只要換development board就能解決。

H. 用 hardware description langugae來完成decoder甚至codec。


spdif
IDE interface
LCD panel
stereo 16/24 bit DAC
Dual 7-Segment Display
Speaker
RS232
Parallel I/O (PIO)
Debugging Timer
Timer
UART
Avalon Bus
sdram controller
ethernet

圖表 11 延伸架構圖

參考資料

[1]Alera,"Using the Nios Embedded Processor, Programmable Logic, IP & SOPC Builder Tools to Construct Custom Microcontrollers" Altera News & Views,pp.7-10,First Quarter 2003

[2]MAD:Mpeg Audio Decoder

http://www.underbit.com/products/mad/

[3]MAD low-level API demonstration

http://www.bsd-dk.dk/~elrond/audio/madlld/

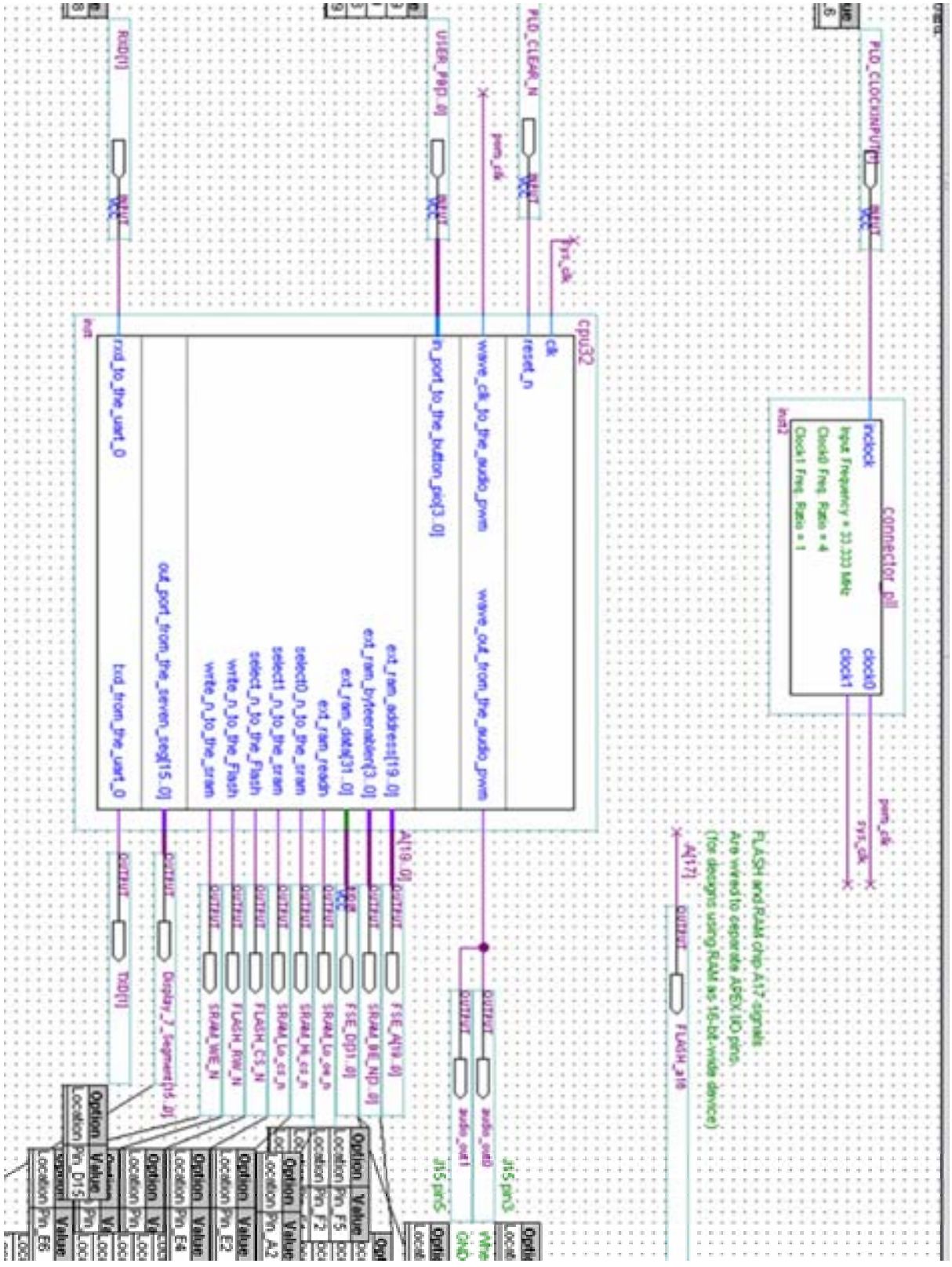
[4] a MP3 player using the Nios development board

http://www.tu-harburg.de/ti6/lehre/soc/ws02/mp3_nios.pdf

[5]蕭如軒,"SOPC系統設計",儒林,2003

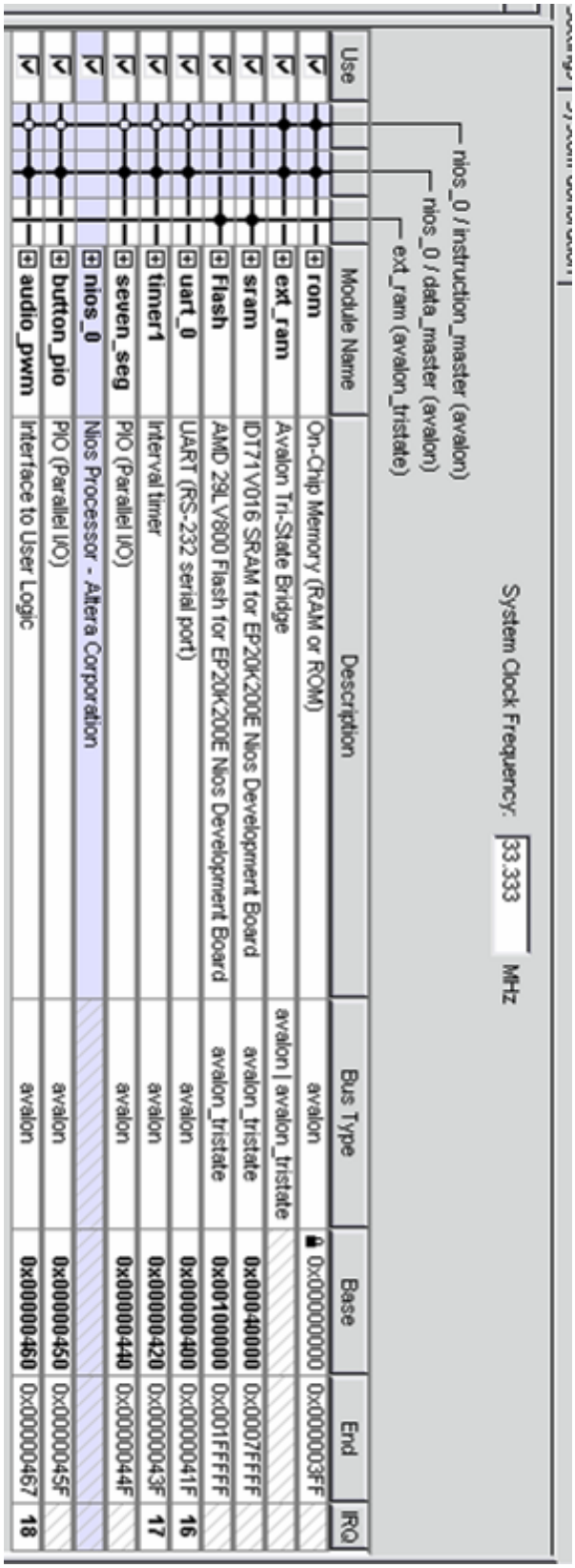
[6]Scot Hacker,MP3: The Definitive Guide,O'Reilly,2000

附錄 A



nios cpu周邊架構

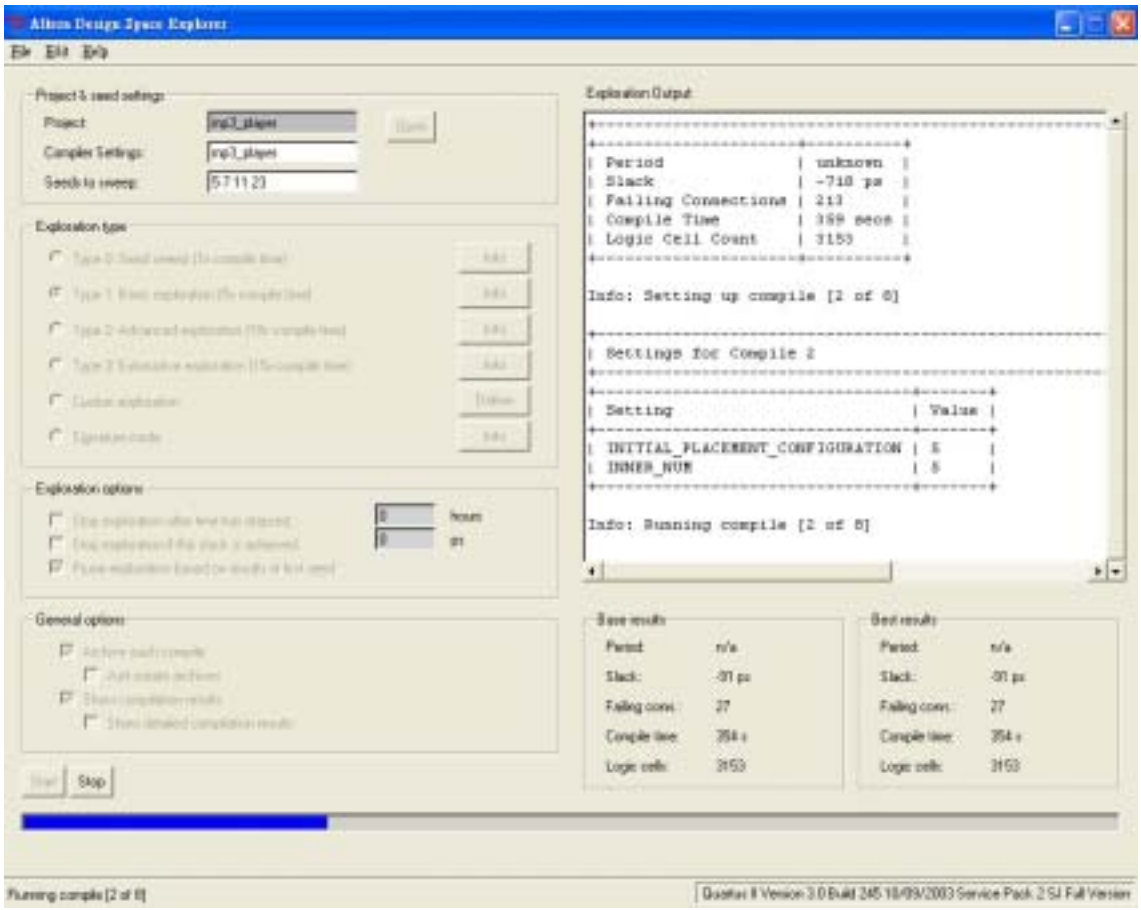
附錄 B



nios Sopc Builder

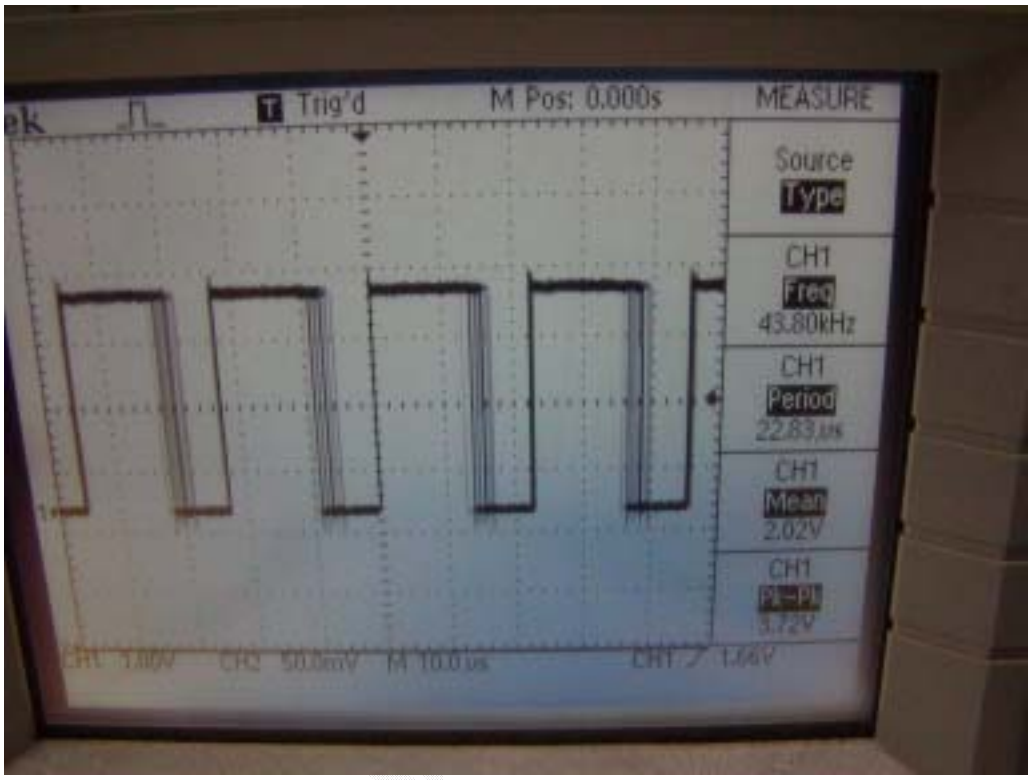
附錄 C

Altera Design Space Explorer操作畫面



附錄 D

聲音輸出波形


Trig'd
M Pos: 0.000s
MEASURE
Source
Type
CH1
Freq
43.80kHz
CH1
Period
CH1
Mean
2.02V
CH1
Pk-Pk