國立清華大學 電機工程學系
實作專題研究成果摘要

8051 Microcontroller–From RTL to FPGA Realization
8051 微控制器–從 RTL 至 FPGA 軟核實作

專題領域：系統領域

組　　別：B280
指導教授：呂仁碩副教授
組員姓名：惠鈞、許育恩
研究期間：2022年2月25日至2022年12月2日止，共9個月

Abstract

In the heart of most modern electronics, there's one microcontroller or multiple microcontrollers, it is everywhere in everyday life. This tells that microcontroller systems are essential for studying system design, and are worth the time to dive deeper into.

We found that implementations of microcontroller systems in undergraduate researches are very common, but few studies the microcontroller system itself. We aim to study microcontroller system by trying to make a microcontroller and implementing it onto an existing image sensor from professor's lab as a controller for the sensor.

In this project, we hope to research on how a microcontroller system works, in particular the *Intel MCS-51* microcontroller architecture, and implementing the MCU onto a FPGA board. The *Intel MCS-51 MCU* or so called *8051* platform is more basic than other widely used platforms like *Arduino* and *ARM Mbed OS*, which is more suitable for studying on microcontroller system itself. We believe by studying the *8051* architecture, we will have better grasp on how a general MCU system works, and better prepare us for developing larger more complex systems in the future.

We managed to successfully edit and simulate an open-source code core for 8051 to working state, and later implement it onto Nexys A7 FPGA board and ZedBoard SoC Development Board. Other than this, we studied and implemented some communication technologies including UART, I2C, and AXI, to prepare for the next stage of this project.

In the next stage of this project, we will try to print out the core onto a chip and to implement it with a sensor module from a professor's lab previous project as a data processor and data transmitter.

摘要

在生活中常見現代電器的核心裡，都有一顆或多顆微控制器；微控制器可說是除了用於複雜電腦系統的大型處理器之外最重要的技術之一。遙控器、洗衣機、微波爐和各種日常用品內基本都會有微控制器來接受、處理、控制機器其他元件，甚至可以說生活離不開微處理器。

大學部學生對於微控制器的實作運用非常常見，但不常會有學生去實際瞭解微控制器內部的運行架構及指令集如何運作。本專題最終目標嘗試實際製作一顆微控制器，並且嘗試將之與實驗室現有影像感應器合並作爲感應器上的控制器使用。

本專題希望藉由研究 Intel MCS-51 MCU 架構及實際使用 OpenCore 開源碼寫入 FPGA 進行測試來瞭解微控制器的運作過程及原理、分析此微控制器系統的指令集，並經由寫 Assembly Language 進行操作。我們成功將基於 Intel MCS-51架構的開源程式碼調整並進行 Simulation，並成功寫入 FPGA 板上。除此之外，為專題下一階段作準備，例如 UART、I^2C、AXI 等技術的研究及實際運用。本專題下一階段準備將8051MCU 與實驗室研發的影像感應系統實作於同一顆晶片上，作爲數據處理及外界傳輸資料的管道。

Intel MCS-51相對 Arduino、ARM Mbed OS 等高階常用的系統來說更加基礎，更適合研究微控制器原理及系統設計。我們認爲經由深入瞭解及研究微處理器架構及運作，可以爲未來設計更加大型的處理器時作準備，是個很好的實作經驗。

Table of Contents

1. Introduction

本專題主要在瞭解並實作一顆基於 Intel MCS-51架構（以下簡稱8051 MCU）的軟核微控制器。我們成功讓此8051MCU 分別在兩塊開發板上正常運作，並讓其可以利用I^2C技術從 Serial External ROM 讀取指令資料。

Intel MCS-51 微控制器是由英特爾公司在1980年發行的微處理機，由於其低功耗、較小尺寸和簡潔的架構，在嵌入式系統、汽車、消費電器等各個領域內為最常用、最流行的微控制器之一。雖然英特爾公司已經于2007年停產8051 MCU，但至今仍有多家公司生產基於8051之上的微處理器。雖說此控制器已有40年的歷史，它仍然在開發者及業界有廣泛的利用。

8051擁有優越的處理性能，相對於32bit ARM 系統的微處理器，8051更加簡潔、基礎，更加適合用於研究微處理機架構。透過較簡易的8051 MCU 來徹底的了解一次 MCU 底層的這些細節，可以讓我們對於系統設計的基礎觀念有更好的掌握。

雖然8051 MCU 位元數較少只有8位元，但其功能性不輸其他更加多位元的處理器。相較於其他市售8051微處理器，本專題8051微處理器不同之處有 [2]：

- 獨立 External ROM Data 接口，可以從 External ROM 讀取 Serial Data。
- 獨立全雙工串口（UART），無需經過 Port 3使用 UART 功能
- 獨立特殊控制寄存器組（SFR）
- FPGA 上 Clock Frequency 可達50MHz
- 內建128字節暫存器（不包含特殊控制寄存器）
- 128字節程序儲存器（ROM）
- 1個 Clock cycle 就是1個 Machine Cycle

除此之外，原本8051微處理器的功能也擁有：

- 8位算術邏輯單元
- 8位纍加器
- 8位寄存器
- 8位數據接口
- 2個16位定時器/計數器
- 8位、11位及16位操作系統

本專題的8051模組按照原本 MCS-51之設計擁有111個指令，全部指令皆有運作。本模組中的特別之處是，指令集與架構的設計，1 Clock cycle = 1 Machine Cycle，而非 Intel 原始設計的12 Clock cycles = 1 Machine Cycle，使它成為一個 Single Clock Cycle 的 MCU。

爲了讓8051模組可以順利跟外界交流，我們選擇將 UART 和I^2C技術運用在模組中，並分別用於與電腦或其他處理器和 ROM 進行數據交流。這兩個技術的特點

就在於都分別只需要兩條數據線，這樣可以大幅度減少8051模組所需要的外接Pin 腳，節省很多實際下線時的面積。

2. Research Methodology

藉由直接閱讀開源的8051 Verilog code，不僅能夠從理論上了解8051的設計架構，包含 Addressing mode、Instruction Set 的設計、UART 的控制等，更能了解如何將理論中的架構，轉換為 RTL code，並且學習作者在撰寫上的技巧與巧思。而在看懂8051 MCU Verilog code 後，為了我們的應用方向，對原始設計作出改良，或是額外增加需要的功能，是本專題的目標。最終目的是藉由實際走過一遍從 RTL code 到下線成實體晶片的完整過程，學習數位積體電路設計流程。

本專題由開始至今大致可以分為3個階段：

(1) 閱讀 OpenCore 8051 Source code

(2) 利用 FPGA 對 OpenCore 8051進行測試與驗證

(3) 對 OpenCore 8051進行改良與增加新功能

目前正進行到(3)階段，下一步將使用 SoC Encounter，繪製 Physical Layout 與 Post-Layout 驗證的工作。

3. Experimental Results

以下內容皆以OpenCore 8051原代碼為準，因此部分內容與Intel 8051有所出入。

3.1. Source code debugging

在測試OpenCore 8051時我們發現在Functional simulation 中會正確運作，但是在Post-Synthesis simulation 時會出錯。在經過我們的測試後發現，是因為作者在原代碼中的case 語句中加入Synthesis 指令 “// synopsys full_case”的緣故，導致在使用Xilinx Vivado 進行合成後會出錯。

若在case 語句中沒有default的選項，綜合器會產生一個latch，因為case都不成立時，暫存器要保持上一次的值。而加入“//synopsys full_case”在case語句後方，告訴綜合器，所有的case已經覆蓋，不需要自動產生latch，以節省硬體資源。但產生的副作用是，設計者要保證只會有case1和case2，不會有case3的出現，否則A會是一個不確定的值。

而在原代碼的設計中，有部分的case 語句在無法保證不出現“case3”的情況下使用“//synopsys full_case”，導致合成出錯，在我們將其刪除後，便可以得出正確的結果。

```
case(case_signal)
begin
    case1: A = B;
    case2: A = C;
end
endcase
```

```
case(case_signal) //synopsys full_case
begin
    case1: A = B;
    case2: A = C;
end
endcase
```

Fig.3.1-1 Full case 範例

3.2. I²C External EEPROM Interface

為了在下線成晶片後能夠修改Instruction ROM的內容，使用External ROM是必須的。但是在原始設計中，與External ROM 溝通的介面由16 bit address[15:0] 和32 bit data[31:0] 組成，高達48 Pin的介面要能夠成功下線顯然是不切實際的。因此我們在原始的External ROM介面上增加一層I²C Interface，該Interface包含4 KB的RAM。在每次開機時，會先經由I²C Interface讀取External ROM地址從0到4096的資料至Interface的RAM中，讀取完成後再使8051 MCU開始運作。藉由這種方式使得從原先的48 Pin 降為僅需I²C的2 Pin(SDA & SCL)即可。

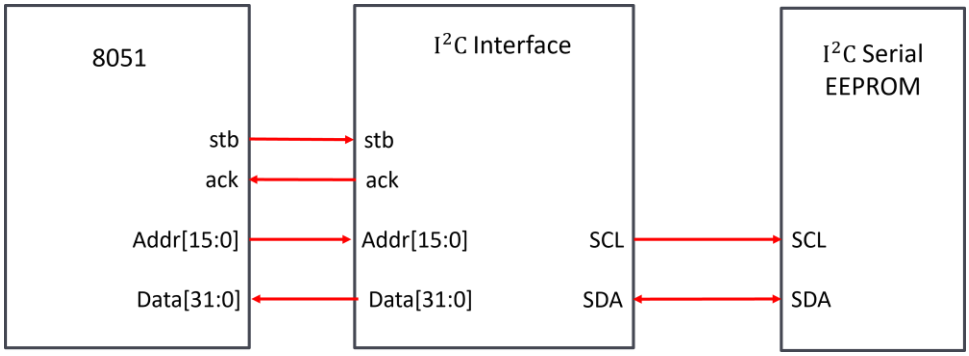


Fig.3.2-1 I²C Interface block diagram

在實作中我們選用 Microchip 24LC64 64kbit I²C Serial EEPROM 作為 External ROM。為模擬下線成晶片後的操作情形，利用 Arduino UNO 將資料寫入24LC64中，再由8051 I²C interface 讀出。

電路圖請見 Fig.3.2-2，其中：

Master 1： Arduino UNO

Master 2：I²C interface

Slave：24LC64

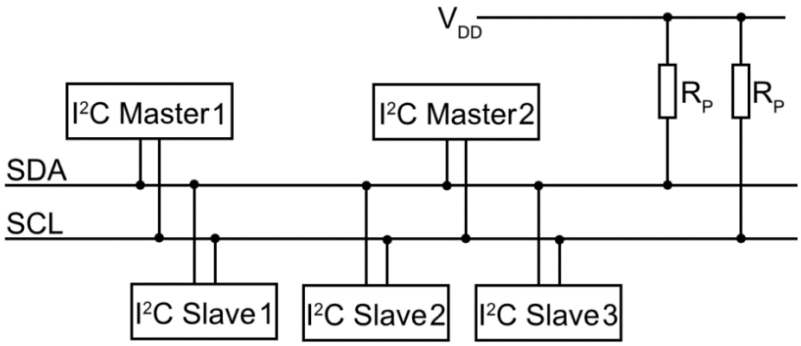


Fig.3.2-2 多個I²C Master對多個I²C Slave

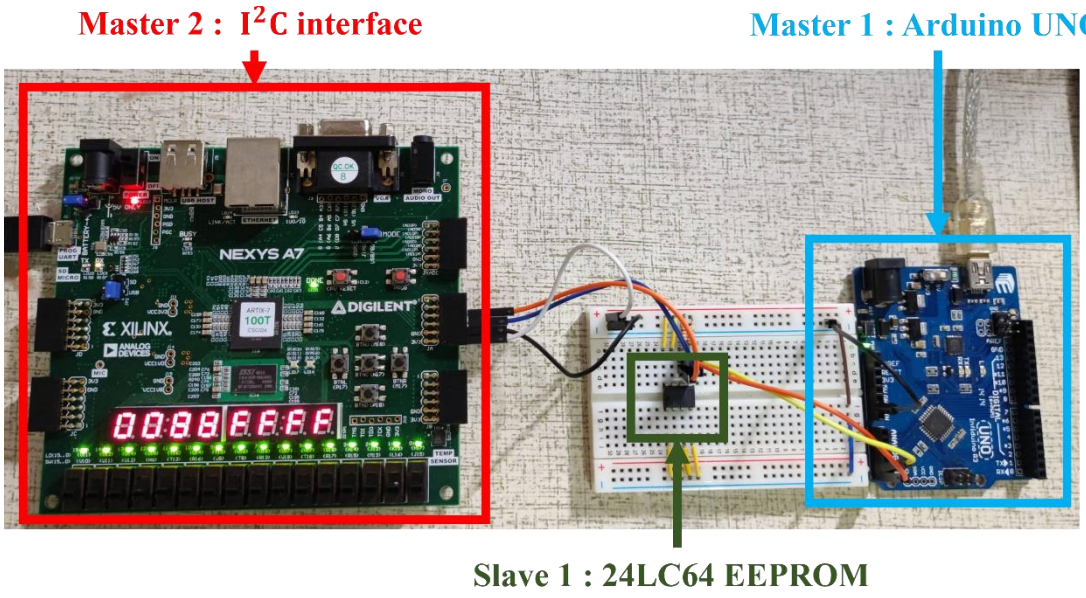


Fig.3.2-3 實際Nexys A7 8051經由I2C讀取External ROM電路圖

3.2.1. I²C interface 撰寫與驗證

I²C interface Verilog 的撰寫部分，參考網路上的資訊與範例[5]後進行修改，使其符合我們使用的24LC64 EEPROM。

從該範例中需要修改的地方有以下兩點：

(1) 1 byte address 改為2 byte address

24LC64需要2 byte 作為地址訊號，因此需要更改範例中的設計的Finite State Machine 使其傳送2 byte address 而非1 byte address

(2) 增加Sequential Read

在範例中僅支援Random read，不支援Sequential read，但是在我們的應用中，僅需要一次的Sequential read，從0讀至4096即可，因此這是可以針對所需的應用而作優化的地方。以下計算兩者所花費的時間差別：

Random read：

讀出1個byte 需要：

$$Control\ +\ addr.(2\ bytes) + Control + data$$
$$T_{byte} = 1 + 2 + 1 + 1 = 5(cycle)$$

因此讀n bytes需要：

$$T_{ran\_total} = 5n$$

Sequential read：

連續讀n bytes：

$$Control\ +\ addr.(2\ bytes) + Control + n * data$$
$$T_{seq\_total} = 1 + 2 + 1 + n = n + 4(cycle)$$

因此在$n \gg 1$時，$T_{ran\_total}/T_{seq\_total} \approx 5$，兩者所需要花的時間差距高達5倍。

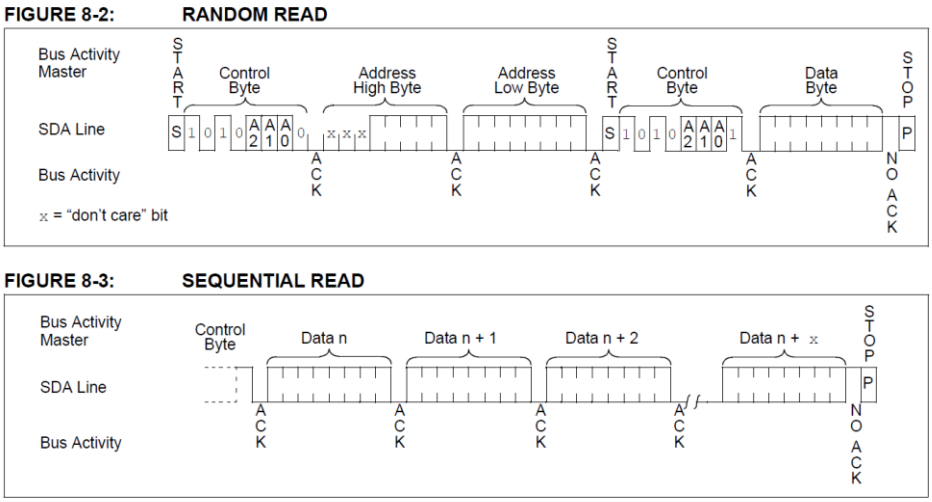
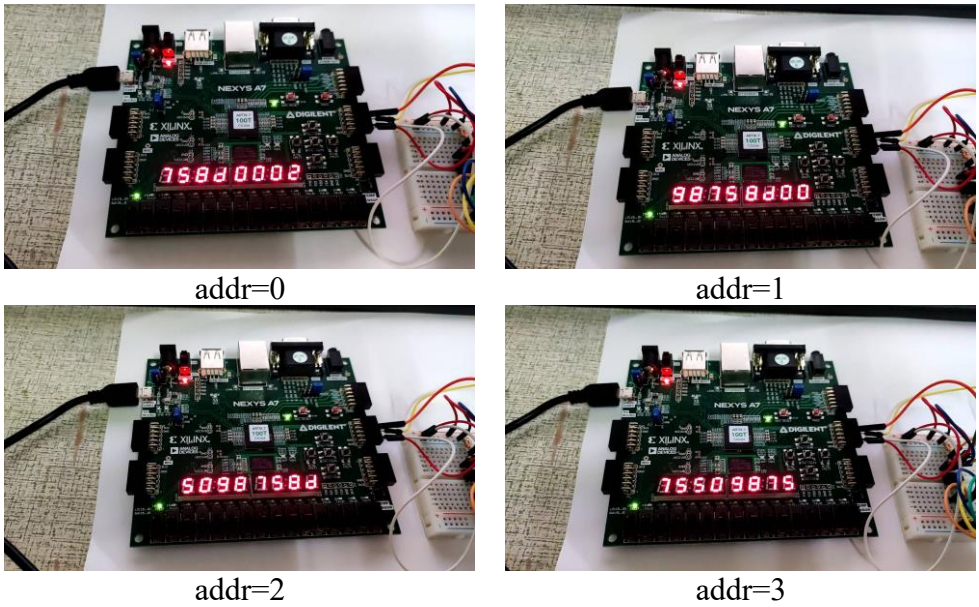


Fig.3.2-4 24LC64 Random read & Sequential read[6]

關於 I²C interface的測試，將經由 I²C從24LC64中讀出的數值存於buffer中，並將其顯示在8個七段顯示器上，由左至右分別顯示buff[addr+3]，buff[addr+2]，buff[addr+1]，buff[addr]，總共顯示4個byte的資料，由Fig.3.2-5可以看出，當addr增加時，七段顯示器中的數值向右移，藉此對比由Arduino UNO 寫入的數據，驗證I²C interface讀取的數值正確。



addr=0　　addr=1

addr=2　　addr=3

Fig.3.2-5 I²C interface測試結果

3.2.2. 結合 I²C interface與OpenCore 8051

利用Arduino UNO將3.4.1 中使用的UART 測試程式寫入24LC64中。使用I²C interface取代OpenCore 8051的External ROM，在開機時8051持續處在reset狀態，只有當I²C interface 讀取完資料後才會enable 8051，使其開始運作。由Fig.3.2-7可以看出8051正確的運行。

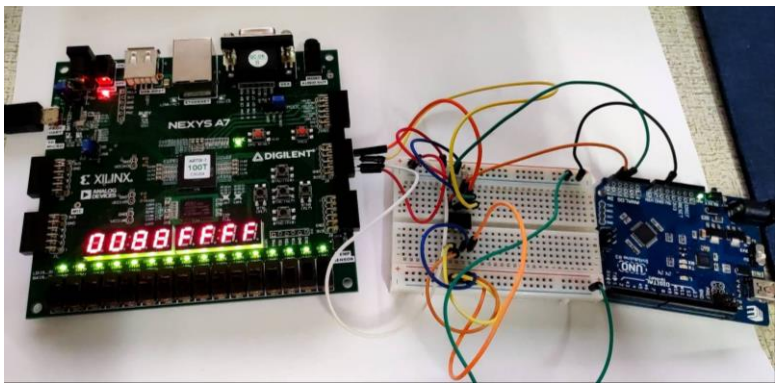


Fig.3.2-6 結合 I^2C interface與OpenCore 8051

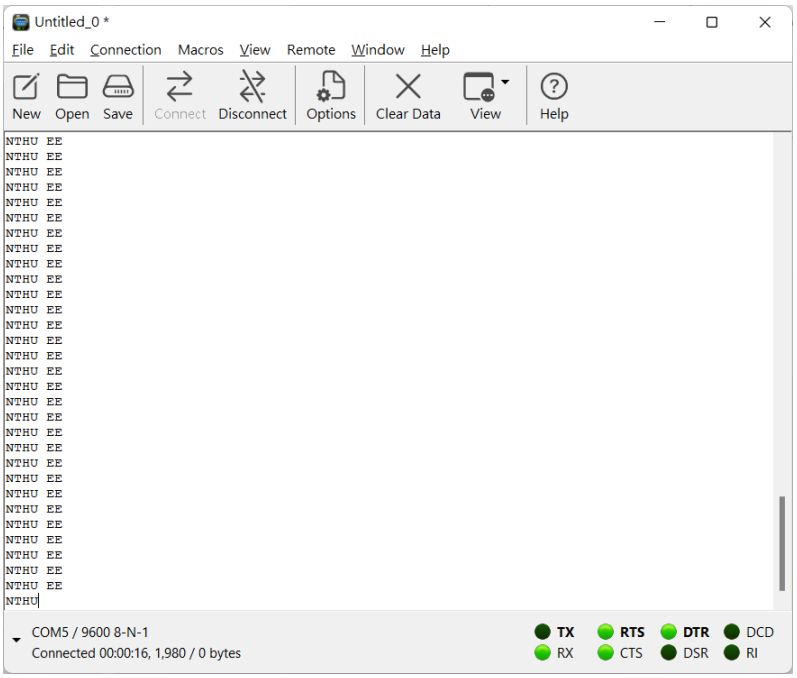


Fig.3.2-7 PC端接收的資料

3.3. 8051 MCU Architecture

藉由閱讀原代碼，整理出OpenCore 8051架構圖Fig.3.3-1，在此架構中，它成為一個Single Clock Cycle 的MCU，而非遵照Intel 設計的12 Clock cycles = 1 Machine cycle。另外與Intel 8051不同的地方為，此架構中的UART、External ROM等接口並不與GPIO共用接腳，因此所需的接腳數增加。

此架構中除了Decoder, ALU等標準模組外，另有一個較特別的模組，memory interface，該模組不僅做為RAM, ROM與其他模組之間的溝通介面，還包含Program Counter(PC) 的控制，代表它會參與到Program branch、Interrupt。

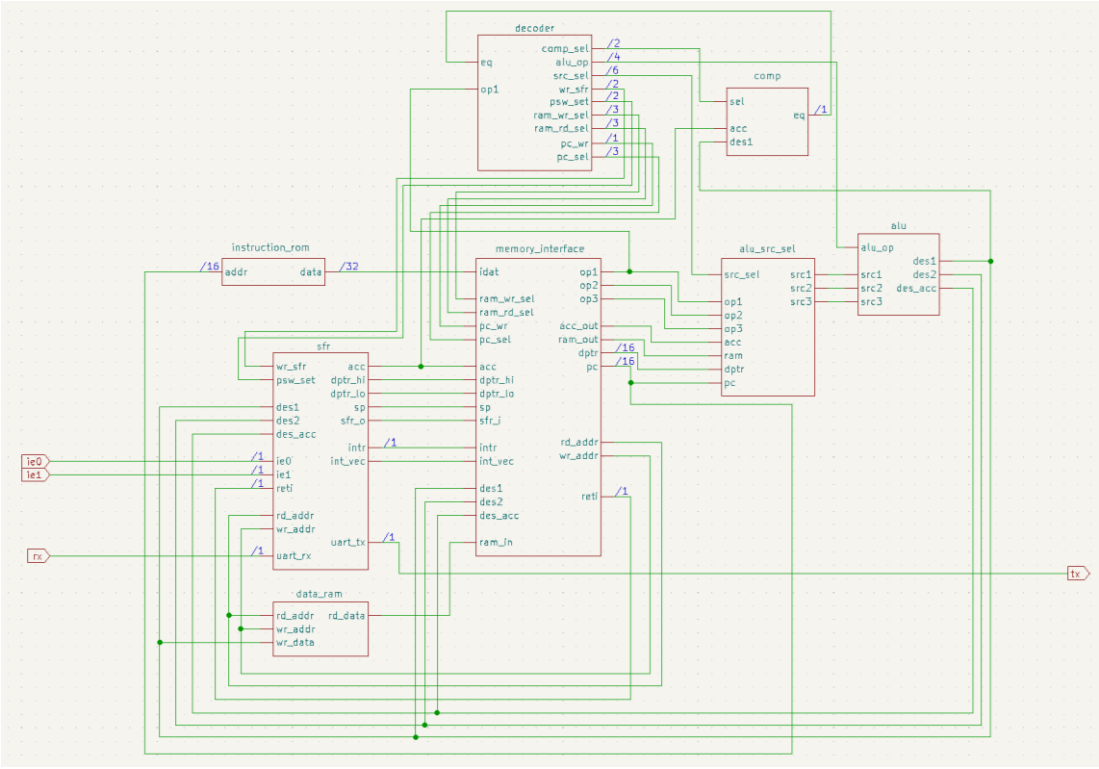


Fig.3.3-1 8051 MCU Architecture

圖中所有未標註的資料線為8 bits。

3.4. UART

3.4.1. NEXYS A7

因為NEXYS A7 開發版中已包含FTDI FT2232HQ USB-UART bridge[4]，因此可以將實做於Artix-7 FPGA 中8051 UART的Rx, Tx 分別接至C4, D4，即可通過Micro-USB接口與PC連接並溝通。

在PC端利用軟體CoolTerm與8051溝通，並撰寫Assembly code 與 C code控制8051 MCU，使其持續傳輸“NTHU EE”字串。測試結果請見Fig.3.4-2，在PC端正確的接收到8051 MCU傳輸的字串。

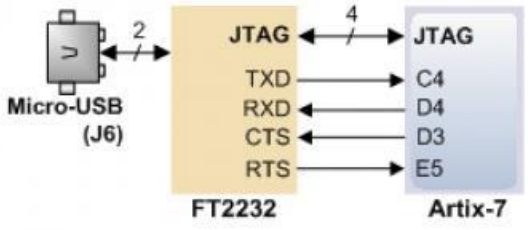


Fig.3.4-1 NEXYS A7 FT2322HQ Connections

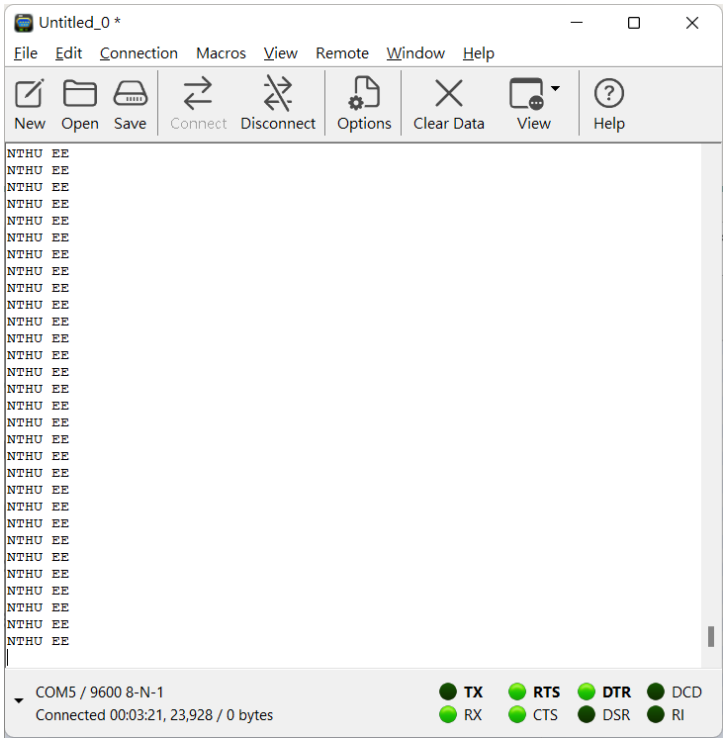


Fig.3.4-2 PC端接收的資料

3.4.2. ZedBoard

ZedBoard SoC Development Board上搭載的是Xilinx Zynq-7000 All Programmable SoC系統，主要可以分成Processing System（PS）跟Programmable Logic（PL）兩個部分。主要構造圖如下：

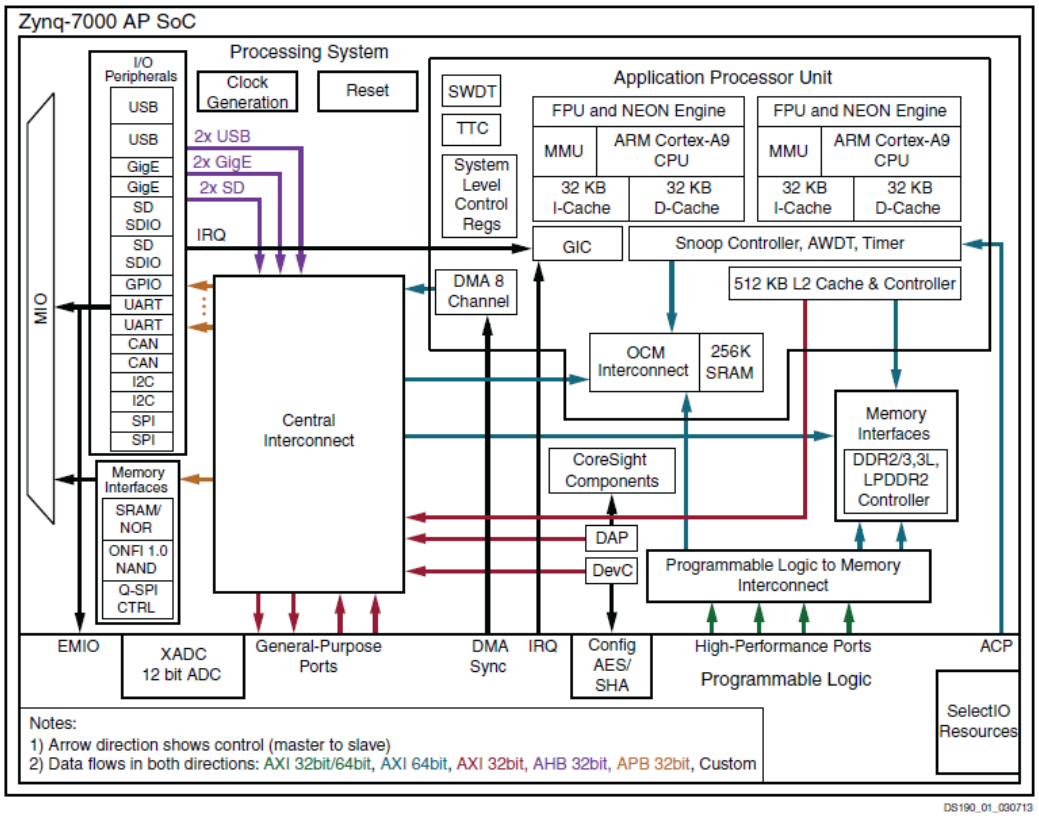


Fig.3.4-3 Zynq 系統

在Zynq系統中搭載著Cypress CY7C64225 USB-to-UART模組，此模組包含在PS系統的I/O Peripherals裏（Fig.3.4-3左上方），我們希望可以將在PL系統中8051軟核的UART訊號經由Extended MIO interface跟Cypress USB-to-UART接口接上，再經由USB-UART模組傳輸至電腦。

在嘗試鏈接PL跟PS I/O之後，由於PS系統設定的兼容性問題無法從Cypress USB-UART傳輸出訊號，一再嘗試無果。由於耗費了長時間在此問題上，同時也因爲當8051下線時將不會有Cypress USB-to-UART模組，我們決定直接繞過PS系統經由PMOD接頭外接USB-UART模組（Digilent PmodUSBUART），並成功從8051軟核傳輸資訊至電腦CoolTerm軟體。

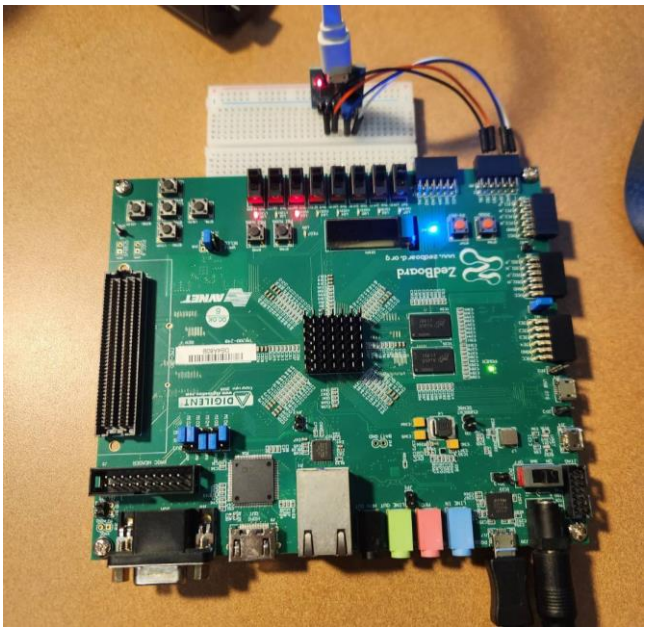


Fig.3.4-4 ZedBoard 8051經由UART與電腦傳輸電路圖

3.5.8051 MCU Instruction Set

3.5.1. Instruction Set

RISC 處理器的Register 數量多，可以讀取多個數據，進行處理後再統一寫回RAM中，而相較於RISC 處理器而言，8051 MCU 的Register 數量較少，通常需要對單一數據處理後便需要寫回RAM 中，因此大部分的8051 Instruction 中，便已經包含從RAM中讀取數值、運算並且寫回的三個步驟。

8051的Instruction 以 byte 為單位，可以是1~3 bytes：
1st byte: operation code(opcode)
2nd and 3rd byte: address or immediate

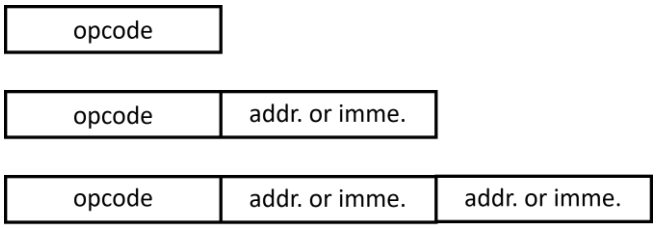


Fig.3.5-1 8051 instruction 示意圖

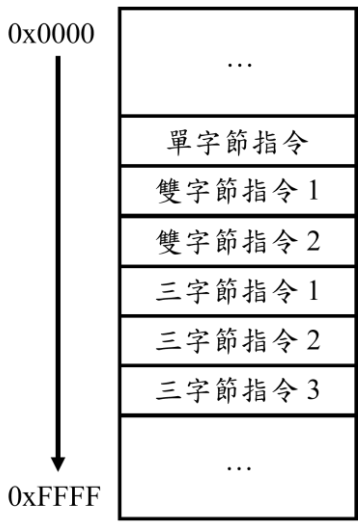


Fig.3.5-2 8051 instruction ROM

總共有111 Instructions, 可以被分為5種：

(1) Data Transfer Instructions
MOV, PUSH, POP, …

(2) Arithmetic Instructions
ADD, SUBB, INC, DEC, MUL, DIV, …

(3) Logical Instructions
ANL, ORL, XRL, CLR, …

(4) Boolean or Bit Manipulation Instructions
ANL, ORL, CLR, …

(5) Program Branching Instructions
ACALL, AJMP, LJMP, …

3.5.2. Addressing mode

8051 Instruction Set 有以下6種 addressing mode。

(1) Immediate Addressing
運算數包含於Instruction 2nd and/or 3rd byte。

(2) Direct Addressing
運算數的地址包含於Instruction 2nd and/or 3rd byte。

(3) Register Addressing
取用被PSW(Program Status Word)選中的Register Bank 中R0-R7所存的數值作為運算數。

(4) Register-Indirect Addressing
取用Register Bank 中R0-R1所存的數值作為運算數的地址。

(5) Indexed Addressing
運算數的地址為base register(PC, DPTR)+index register(ACC)。

(6) Bit Addressing
用於bit addressable的區塊，指定其中1個bit 進行讀寫。在8051的RAM中地址

為 20H~2FH、SFR 區中以 0 or 8 結尾的byte為 bit addressable。
在RAM的8 bit address 中：

[7] =
0: 20H~2FH (16 bytes)
1: SFR (80H~FFH中以 0 or 8 結尾的16 bytes)
[6:3] = byte selection
[2:0] = bit selection

舉例而言：
MOV C, 31H;

31H = 0011_0001

地址為26H的第1個bit。因此該指令會將地址為0x26中的第1個bit 寫入PSW的CY中。

4. Conclusion

本專題中，藉由原代碼的閱讀，熟悉8051 MCU的ISA、Architecture 等。而在實作的部分成功擴充電路，使8051能夠以外部I^2C ROM開機，是下線晶片時必要，但是8051缺乏的。另外藉由實際對於RTL 的一連串的模擬，包括Functional, Post-Synthesis, Post-Implementation Simulation，使我們對於RTL在FPGA的驗證，有充足的練習，因此對於移植RTL到不同的FPGA更有信心。預期接下來進行數位積體電路設計流程的後端部分，使用APR(Automatic Placement & Routing)軟體，SoC Encounter，製作Physical Layout，並進行Post-Layout Verification。

5. Reference

[1] 李新兵著，《8051軟核處理器設計實戰》，機械工業出版社，2015年3月第一版

[2] Jaka Simsic, Simon Teram,《8051 core》原始碼，https://opencores.org/projects/8051，2016年11月13日更新版本

[3] Louise H. Crockett, Ross A. Elliot, Martin A. Enderwitz, Robert W. Stewart,《The Zynq Book》, 1st Edition, Strathclyde Academic Media, Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, Scotland, UK

[4] Nexys A7 Reference Manual, https://digilent.com/reference/programmable-logic/nexys-a7/start, Digilent, July 10 2019

[5] Yongxiang-Guo, Verilog_i2c_eeprom, https://github.com/Yongxiang-Guo/Verilog_i2c_eeprom

[6] 24LC64 datasheet, Microchip, https://www.microchip.com/en-us/product/24LC64

6. Plan Management and Teamwork

教授在本專題期間主要提出專題題目及大略方向，並且引導學生往此方向前進。每周專題生及教授將會開一次會議，專題生在此會議中會做出進度報告及提出本星期遇到的問題。教授則會針對報告進行提問及嘗試回答學生的問題或提出解決方案讓學生嘗試。

教授的引導方向是希望利用反向教學的方式，由我們教老師，使我們需要更紮實並清楚的了解的問題成因與解決辦法，藉此使我們獲得充實的知識。

隊友合作方面，若隊友有遇到瓶頸，另一位隊友就會嘗試幫忙解決；若一切順利沒有問題則各自負責各自所分配的部分。

初期，前三個月（設置及跑模擬）:

- 許育恩及惠鈞：

  共同合作先行分析，並將開源碼編輯成完整且可以模擬的完整核心碼。完成之後，再包裝成可以和其他 IP 模組一起運用的完整碼，並測試完整碼性能。

中期（寫入 Nexys A7 FPGA）:

- 許育恩：

  主要負責將8051 MCU 寫入 Nexys A7 FPGA

- 惠鈞：

  主要負責分析8051指令集，將所有運作細節學習並報告給其他人學習、繪製8051 架構圖。

- 合作：

  中間有部分時間嘗試解決8051模組 Post-synthesis 的 Error，兩位同學一起合作嘗試解決。

後期（嘗試使用 ZedBoard SoC Development Board 及 EEPROM）:

- 許育恩：

  主要負責將8051 MCU 寫入 ZedBoard，

- 惠鈞：

  8051 UART 等功能驗證、使用I^2C技術讓在 FPGA 上的8051核心模組從外接 EEPROM 上讀取指令資料

- 合作：

  報告、海報的撰寫

7. Reflection

在本次專題進行過程中，曾多次碰到瓶頸，但在嘗試解決問題的時候，學到了相當多的相關知識，尤其是對於8051的架構、FPGA有更全面且深入的了解。藉由將8051在不同FPGA版中實作的經驗，我們對於如何將RTL移植到FPGA上有充分的經驗，以及十足的信心。實際對於RTL 的一連串的模擬，包括Functional, post-synthesis, post-implement simulation，使我們對於RTL的驗證，有充足的經驗。另外我們成功擴充電路，使8051能夠以外部I^2C ROM開機，是下線晶片時必要，但是8051缺乏的。

在最後階段時嘗試使用Zedboard開發板的內建USB-UART模組從PL（8051），經由EMIO UART（PS）將訊息傳輸到電腦上，嘗試相當多方法解決。雖然最後因最後下線時將不會有USB-UART模組，而決定直接使用PMOD接頭連UART，但在途中我們對於Zedboard與Zynq SoC體系有更深入的瞭解，對System on Chip的原理和使用經驗增加。

我們認爲稍微可惜的一點是由於研究的性質，我們不需要讀太多的文獻，少去了不少看其他工程師或學生解決問題及研究的方式。除此之外還有一些可以改進的地方，最重要的就是一開始對於階段目標的設定，因爲沒有階段目標而造成時間規劃比較混亂，若有明確設立階段目標的話，相信我們有機會完成更多的部分。

在這兩個學期中我們非常感謝教授的耐心教導，領導我們從瞭解8051一直到成功將它寫入FPGA，我們期待接下來繼續與教授合作完成此專題。