2110362 Microcomputer Interfacing Laboratory Manual

First Semester, 2003

Department of Computer Engineering
Chulalongkorn University

ปรับปรุง ภาคต้น 2546

2110362 Microcomputer Interfacing Laboratory Manual

First Semester, 2003

Department of Computer Engineering
Chulalongkorn University

กิตติกรรมประกาศ

ในสมัยที่ผมเรียนปริญญาตรีที่ภาควิชาฯ นิสิตต้องเรียนภาษา assembly บนไมโครโปรเซสเซอร์ตัวเล็กๆเช่น 6800, 8051 ที่เป็นที่นิยมในขณะนั้น ในโลกของคอมพิวเตอร์สิ่งต่างๆล้าสมัยไปอย่างรวดเร็ว สิ่งที่เรียนมาบางอย่างก็ยังพอใช้ได้ บางอย่างก็ไม่มีใครพูดถึงกันแล้ว วิชา MICRO INTF LAB นับได้ว่าเป็นความภาคภูมิใจอย่างหนึ่งของภาควิชาฯ เนื่องจาก FPGA ยังเป็นเทคโนโลยีที่ใหม่มากเมื่อครั้งเปิดการสอนวิชานี้ในปีแรก กว่าจะมาเป็นวิชา MICRO INTF LAB ให้พวกเราได้เรียนกัน อาจารย์หลายท่าน ได้ทุ่มเทความพยายามและเวลาลงไปเป็นจำนวนมาก คุณราชพร (นิสิตป.เอก) ช่วยออกแบบ FPGA board และสั่งซื้อของ นิสิตป.ตรีใช้เวลาหลายวันเพื่อบัดกรี FPGA board นิสิตป.โทและป.เอกที่ช่วยสอนในวิชานี้ในปีแรกๆ ก็ต้องพยายามอย่างมาก เนื่องจากเป็นวิชาใหม่ จะซักถามคนอื่นก็ยาก ต้องลองผิดลองถูกและพึ่งตนเองอยู่ตลอดเวลา ขอขอบคุณอาจารย์ฐิต อาจารย์บุญชัย อาจารย์ประภาส ที่สนับสนุนวิชา MICRO INTF LAB มาโดยตลอด ไม่ว่าจะเป็นการเอื้อเฟื้องบประมาณ สถานที่ ให้นิสิตเข้าร่วมการแข่งขันการออกแบบวงจรรวม ที่ NECTEC จัดขึ้นสองครั้ง ผู้ช่วยสอนกลุ่มแรกๆก็มาจากนิสิตที่เข้าแข่งขัน คุณประพนธ์ (เรียนโทจบไปแล้ว ทำงานออกแบบฮาร์ดแวร์อยู่กับบริษัทเอกชน) อาจารย์เกริก (ศึกษาต่อต่างประเทศ) นิสิตที่ช่วยสอนในปีนี้มีสามคนคือ มีน โจ้ และผม

ขอขอบพระคุณอาจารย์ทุกท่าน และเพื่อนๆทุกคน

ชัชวิทย์ อาภรณ์เทวัญ
ภาควิชาวิศวกรรมคอมพิวเตอร์
21/5/2546

สารบาญ

สารบาญ (ต่อ)

Page

All parts should go together without forcing.

You must remember that the parts you are reassembling

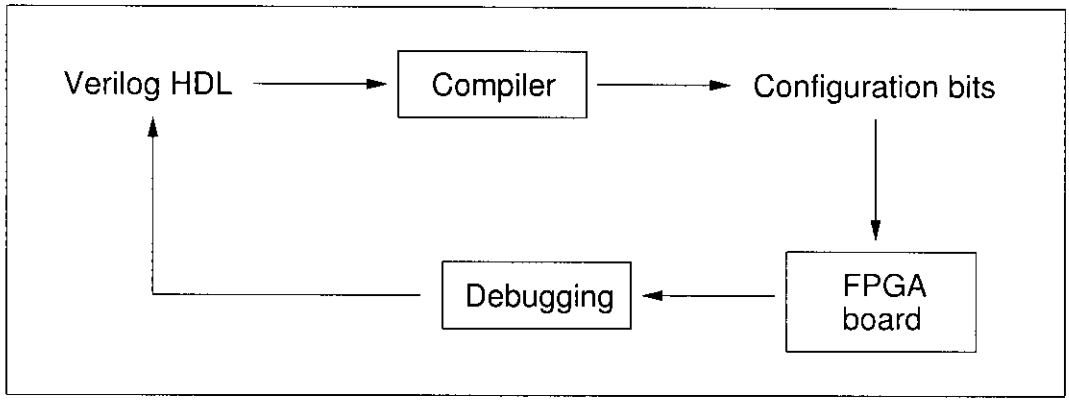
were disassembled by you. Therefore, if you can't get them together again,

there must be a reason. By all means, do not use a hammer.

(IBM maintenance manual, 1925)

บทที่ 1
บทนำ

ในวิชานี้ นิสิตจะต้องใช้ Field-Programmable Gate Arrays (FPGAs) ข้างใน FPGA ประกอบด้วย RAM, functional units, และสายไฟจำนวนมาก เราสามารถโปรแกรมให้สายไฟต่อเชื่อมกันได้อย่างที่เราต้องการ โดยรูปแบบการต่อเชื่อมของสายไฟจะถูกกำหนดด้วย configuration bits ที่อยู่ใน RAM นิสิตจะต้องทำ lab ทุกสัปดาห์ มีการสอบปฏิบัติในกลางภาคการศึกษา และต้องทำ project 1 ชิ้น การทำ lab จะมีรูปแบบดังนี้



การคอมไพล์, ดีบั๊ก, และแก้โค้ด 1 ครั้ง จะใช้เวลา 5-10 นาที (ขึ้นอยู่กับความซับซ้อนของ Verilog และประสิทธิภาพของเครื่องคอมพิวเตอร์ที่ใช้) ถ้าการคอมไพล์ 1 รอบใช้เวลา 10 นาที ในการทำ lab 3 ชม. นิสิตจะมีโอกาสคอมไพล์แค่ 18 ครั้งเท่านั้น ดังนั้นจะใช้นิสัยแบบการเขียนโปรแกรมบน PC ไม่ได้ เช่น เขียนๆไปก่อน คอมไพล์ แล้วค่อยมาแก้ error ทีหลัง เนื่องจากการคอมไพล์ Verilog HDL แต่ละครั้ง ใช้เวลานานมาก นิสิตควรจะมีความแม่นยำในการเขียน Verilog HDL และมีความชำนาญในการใช้ FPGA board คู่มือเล่มนี้เขียนขึ้นเพื่อช่วยให้นิสิตให้ทำ lab ได้ดีขึ้น โดยรวบรวมเนื้อหาต่างๆที่นิสิตต้องรู้เพื่อที่จะทำ lab ให้ได้ภายในเวลาจำกัด นอกจากนี้ยังรวบรวมข้อผิดพลาดที่มักจะเกิดขึ้นบ่อยๆ นิสิตควรอ่านคู่มือและเตรียมตัวก่อนมาทำ lab ทุกครั้ง

A language that doesn't affect the way you
think about programming is not worth knowing.

บทที่ 2

Verilog HDL

ในการทำ lab เราจะเขียน Verilog HDL แบบง่ายๆไม่ซับซ้อน เพราะต้องคอมไพล์เป็นฮาร์ดแวร์จริงๆ คำสั่งที่เป็น high-level มากๆ เช่น for-loop จะใช้ได้เฉพาะบน simulator เท่านั้น โจทย์ทุกข้อใน lab สามารถทำได้โดยการเขียน finite-state machine ตัวใหญ่ๆ ตัวอย่างของการเขียน FSM ที่คอมไพล์ เป็นฮาร์ดแวร์ได้

```
always @(posedge clock)
  begin
    if (reset == 0)
      begin
        // initialize state and some registers
      end
    else
      begin
        case (state[1:0])
          2'b00 : begin
                    // do something
                    // set current state to next state
                  end
          2'b01 : begin
                    // do something
                    // set current state to next state
                  end
        endcase
      end
  end
```

ตัวอย่าง module

```
module ram(cs, oe, wr, addr, data);

  input cs, oe, wr;
  input [16:0] addr;
  inout [7:0] data;

  // do something
endmodule
```

การประกาศตัวแปร

```
wire a, b, c;
wire [15:0] addr;
reg p, q, r;
reg [7:0] data;
```

การกำหนดค่าให้ตัวแปร

```
assign addr[15:0] = 16'b0000_0001_0010_0011;
data[7:0] = 8'b0000_1111;
data[7:0] = ~data[7:0];              // not
data[7:0] = data[7:0] + 1'b1;        // increment
data[7:0] = {data[6:0], data[7]};    // rotate left
```

ที่มักจะเขียนกันผิดก็คือ bidirectional port ให้เขียนแบบนี้

```
module module_name(.., .., .., data_bus);
```

```
... ... ...
inout [7:0] data_bus;

reg ena;
reg [7:0] data_reg;

assign data_bus[7:0] = \
    (ena == 0 ? data_reg[7:0] : 8'bzzzz_zzzz);

// do something
endmodule
```

ถ้าเป็นจังหวะที่เราจะอ่านข้อมูลเข้ามาใน module ก็ให้ ena = 1 ถ้าเป็นจังหวะที่เราจะเขียนข้อมูลออกไปนอก module ก็ให้ ena = 0 ค่าที่จะเขียนออกไปต้องอยู่ใน data_reg[7:0]

การทำ lab ก็จะวนเวียนอยู่กับคำสั่งเหล่านี้ และไม่มีอะไรซับซ้อนไปกว่าการเขียน FSM ตัวใหญ่ๆ

A debugged program is one for which

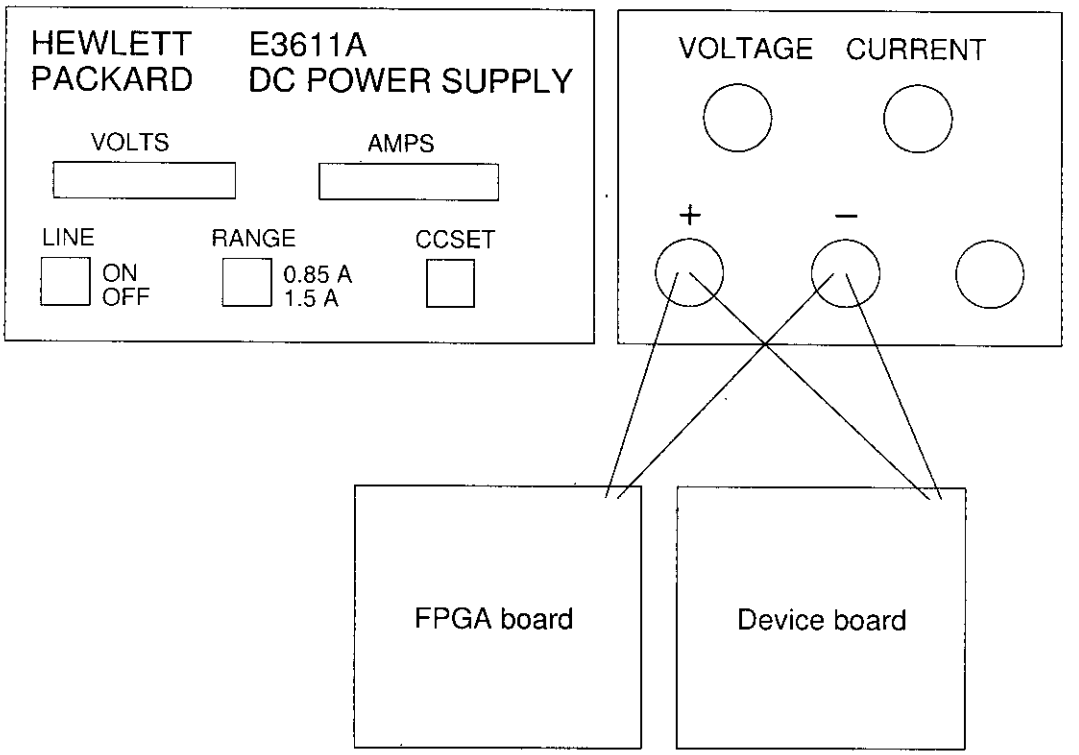
you have not yet found the conditions that make it fail.

(Jerry Ogdin)

บทที่ 3

Power supply

ในการทำ lab ให้ใช้ DC power supply ตัวนี้



กำหนดให้จ่ายไฟ 7.0V ไม่เกิน 1.5A การตั้งค่าให้ทำดังนี้

1. ถอด FPGA board และ devices board ออกจากเครื่องจ่ายไฟ
2. เปิดเครื่องจ่ายไฟ
3. ปรับ RANGE ให้เป็น 1.5A
4. กด CCSET ค้างไว้
5. หมุนปรับ VOLTAGE และ CURRENT ให้เป็น 7.0V และ 1.5A ตามลำดับ
6. ปิดเครื่องจ่ายไฟ
7. ต่อ FPGA board และ devices board ตามเดิม

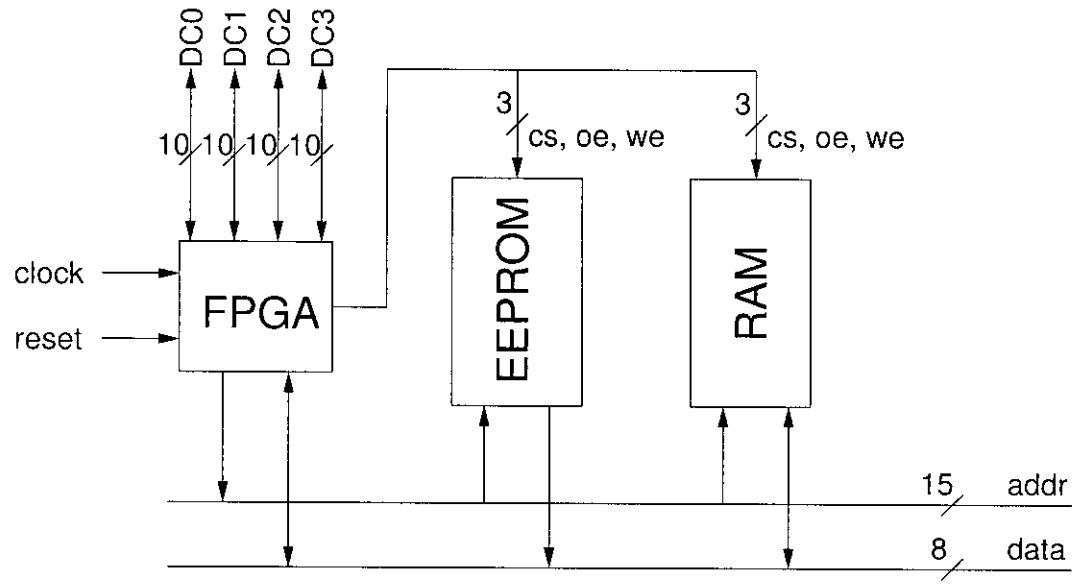
Those parts of the system that you can hit with a hammer (not advised) are called hardware; those program instructions that you can only curse at are called software.

(Levitating Trains and Kamikaze Genes: Technological Literacy for the 1990's)

บทที่ 4

FPGA board

แผนผังของ FPGA board



4.1 FPGA

FPGA ที่ใช้เป็นของ Xilinx รุ่น SPARTAN, S20VQ100, speed grade = -3 (ดู http://www.xilinx.com) เป็น FPGA ขนาดเล็ก แต่ก็ใช้ทำ 8-bit microprocess แบบ RISC ที่มีจำนวน instructions น้อยๆได้

4.2 RESET

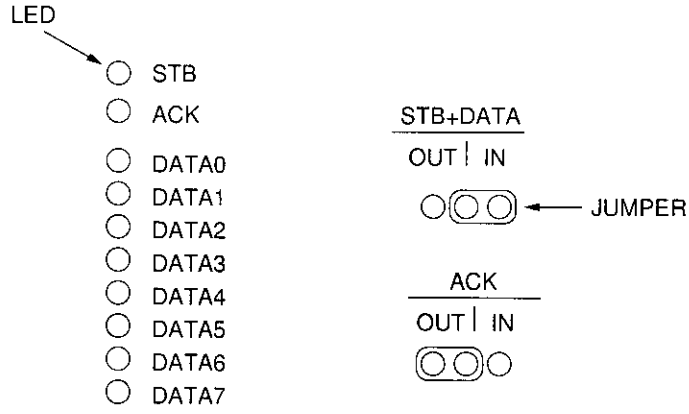
ปุ่มสีดำ ต่ออยู่กับขา 20 ของ FPGA ถ้ากดจะให้ค่าเป็น 0 ถ้าปล่อยจะให้ค่าเป็น 1

4.3 CLOCK

Oscillator (ถังสี่เหลี่ยมสีเงิน) ให้ความถี่ 8 MHz ต่ออยู่กับขาที่ 27 ของ FPGA, socket มันหลวมแล้ว ให้กด oscillator ลงไปบน socket ให้แน่น อย่าพยายามดึง oscillator ออกมาจาก socket

4.4 DC0, DC1, DC2, DC3

DC0 – DC3 เป็น bidirectional port ที่ต่ออยู่กับ FPGA, port นี้จะเป็น input หรือ output ก็ได้ ขึ้นอยู่กับการเซต jumper



ใน 1 port จะมีสายไฟ 10 เส้น คือ STB, ACK, และ DATA0 – DATA7 จะมี jumper 2 ตัวต่อ 1 port, jumper ตัวแรกจะควบคุม STB และ DATA ให้เป็น input หรือ output, jumper ตัวที่สองจะควบคุม ACK ให้เป็น input หรือ output ในตัวอย่างเป็นการเซต jumper ให้ STB+DATA เป็น input และ ACK เป็น output ในแต่ละ port จะมี LEDs 10 ตัว เพื่อแสดงค่าที่อยู่บน port ค่า 0 จะทำให้ LEDs สว่าง และค่า 1 จะทำให้ LEDs ดับ ขอให้เซต jumper ด้วยความระมัดระวัง ตรวจสอบให้แน่ใจว่าเราใช้เป็น input port หรือ output port, port ทุกตัวต่อเชื่อมกับขาของ FPGA ดังนี้ (D7 เป็น MSB)

	DC0	DC1	DC2	DC3
STB	21	79	99	2
ACK	71	78	98	3
DATA0	70	80	90	4
DATA1	69	81	91	5
DATA2	68	82	92	6
DATA3	67	83	93	7
DATA4	66	84	94	8
DATA5	65	85	95	9
DATA6	62	86	96	10
DATA7	61	87	97	13

4.5 EEPROM

FPGA board ถูกออกแบบมาให้ใช้กับ ROM ขนาด $2^{16} \times 8$ บิท แต่เราใช้ EEPROM ขนาด $2^{15} \times 8$ บิท ทำให้ address bus ของ EEPROM กับ RAM ทับซ้อนกันไม่สนิท (แต่ data bus ยังใช้ร่วมกัน) EEPROM ต่อเชื่อมกับ FPGA ดังนี้

EEPROM	FPGA
cs	16
oe	14
we	59
a0	39
a1	40
a2	41
a3	42
a4	43
a5	44
a6	45
a7	46
a8	47
a9	53
a10	55
a11	56
a12	57
a13	58
a14	60
d0	28
d1	29
d2	30
d3	31
d4	32
d5	33
d6	34
d7	35

ดู data sheet ของ EEPROM ได้ที่ http://161.200.93.31/~u37cap/lab/eeprom.pdf

4.6 RAM

EEPROM กับ RAM ใช้ data bus ร่วมกัน ดังนั้นถ้าจะใช้ EEPROM ก็ต้อง disable RAM ถ้าจะใช้ RAM ก็ต้อง disable EEPROM, RAM ต่อเชื่อมกับ FPGA ดังนี้

RAM	FPGA
cs	17
oe	14
we	15
a0 – a13	เหมือนกับ EEPROM
a14	59
d0 – d7	เหมือนกับ EEPROM

ดู data sheet ของ RAM ได้ที่ http://161.200.93.31/~u37cap/lab/ram.pdf

A successful [software] tool is one

that was used to do something undreamed of by its author.

(S. C. Johnson)

บทที่ 5

Compiler

Compiler ที่เราใช้แปลงภาษา Verilog เป็น configuration bits คือ Foundation Series Software v2.1i ให้เปิดโปรแกรม Project Manager ทำตามขั้นตอนต่อไปนี้

1. เลือก Create a new project
2. ตั้งชื่อ project, project จะถูกเก็บไว้ที่ c:/fndtn/active/projects/project_name/
3. เลือก HDL (ไม่ใช่ Schematic), สร้าง new project
4. ใน Design Entry กด HDL Editor, เลือก Create Empty
5. เขียน Verilog, แล้ว Save as type: เป็น Verilog Files (ไฟล์ที่มีนามสกุล .v)
6. save ลงไปที่ c:/fndtn/active/projects/project_name/
7. ในเมนู เลือก Project / Add Source File(s)
8. เพิ่มไฟล์ที่เขียนในขั้นที่ 5 ลงไปใน project
9. กำหนดขา input/output ของ design กับขาของ FPGA เช่น ใน design มีขา input ชื่อ reset เราต้องการให้ขา reset เป็นขาที่ 20 (ต่ออยู่กับปุ่ม reset) ให้เพิ่มบรรทัดต่อไปนี้ลงใน c:/fndtn/active/projects/project_name/project_name.ucf

    NET reset LOC=P20;

ในกรณีที่เป็น array ให้เขียนแบบนี้

    NET data<0> LOC=P28;

10. บนหน้าจอ กด Synthesis (อยู่ข้างล่าง Design Entry)
11. เลือก module ที่จะเป็น Top level (ใน 1 project อาจจะมีมากกว่า 1 module)
12. เลือก Target device เป็น Family=SPARTAN, Device=S20VQ100, Speed=-3, กด Run
13. บนหน้าจอ กด Implementation (อยู่ข้างล่าง Synthesis), กด Run
14. บนหน้าจอ กด Programming (อยู่ข้างล่าง Synthesis) เลือก Hardware Debugger
15. ตรวจสาย download (X-checker) ว่าต่อกับ FPGA board หรือไม่
16. กดปุ่ม reset (สีเหลือง) บน FPGA board

17. ในเมนู เลือก Download / Download Design

Compiler จะรายงานความถี่สูงสุดที่วงจรยังทำงานได้ถูกต้อง อย่างไรก็ตาม วงจรทั่วๆ ไปจะทำงานได้ที่ความถี่มากกว่า 1 MHz บน FPGA นิสิตอาจจะใช้ความถี่ 1 MHz สำหรับทุกๆวงจร, ในโปรแกรมมี features ต่างๆมากมาย นิสิตจะใช้หรือไม่ใช้ก็ได้ ขอให้ทำ lab ให้ทันภายในเวลาที่มีจำกัด คู่มือการใช้ซอฟต์แวร์ให้ค้นหาจาก http://toolbox.xilinx.com/docsan/3_1i/data/fndtn/fsu/fsu.htm

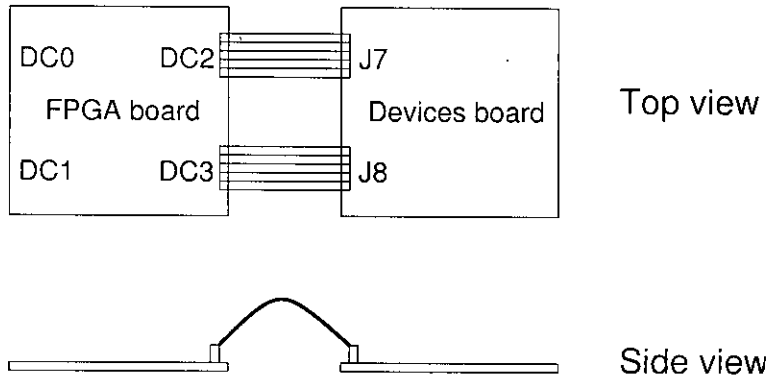
All laws are simulations of reality.

(John C. Lilly)

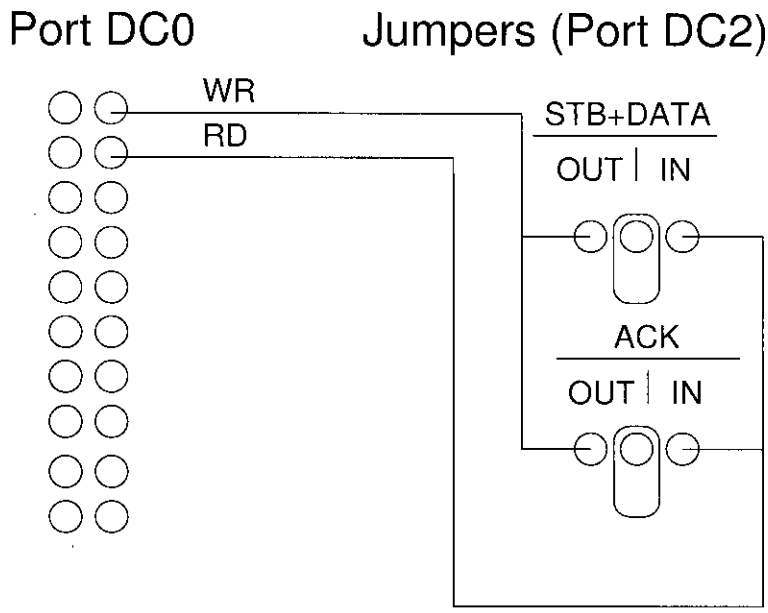
บทที่ 6

Devices board

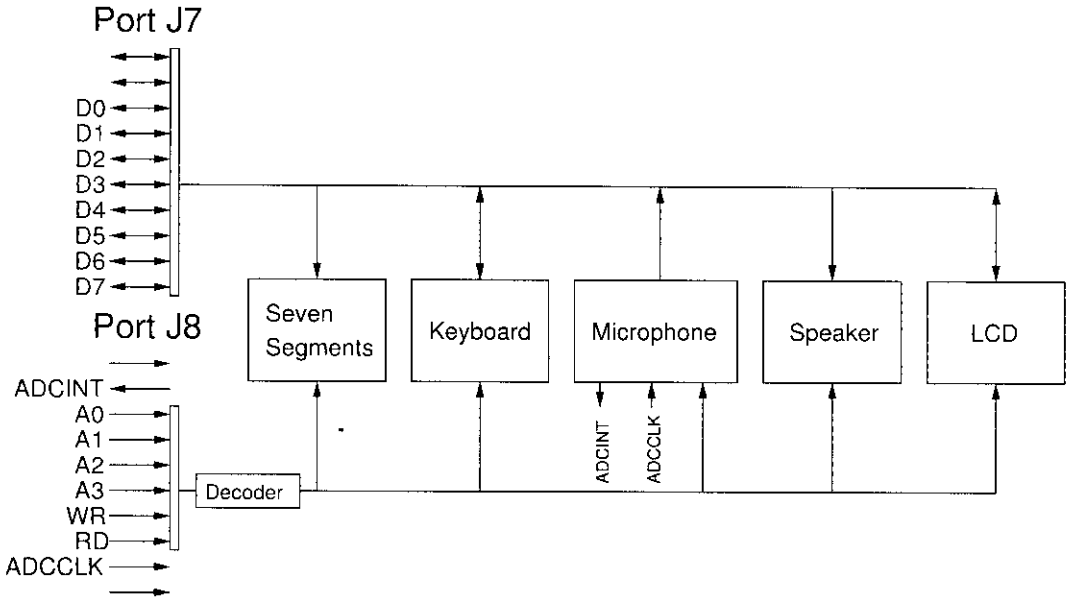
ใน devices board จะมีอุปกรณ์ต่างๆให้ใช้มากมาย เช่น 7-segments, keyboard, microphone, speaker และจอ LCD นิสิตต้องสร้างสัญญาณจาก FPGA เพื่อควบคุมให้อุปกรณ์ต่างๆ ทำงานได้ถูกต้อง ดูวิธีการเชื่อมต่อ FPGA board กับ devices board



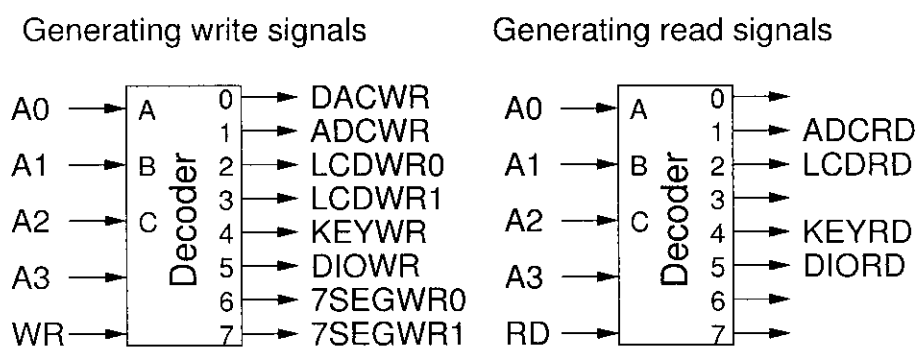
Port J7 เป็น data bus ขนาด 8 บิท ที่เชื่อมระหว่าง FPGA board กับ devices board, FPGA ต้อง read/write ข้อมูลผ่าน port นี้ แต่การควบคุม port DC2 ขึ้นอยู่กับการเซต jumper ดังนั้นเราจะแก้ไข FPGA board เล็กน้อย ดูรูปข้างล่าง



ขาตรงกลางของ jumper เป็นขา ground ดังนั้นถ้าจะเซต jumper เป็น output port ก็ต้องส่ง 0 มาที่ขา OUT และส่ง 1 มาที่ขา IN ของ jumper, WR = ~RD เสมอ, port DC3 ให้เซต jumper เป็น output port เสมอ เพราะ port J7 รับสัญญาณไปควบคุมอุปกรณ์บน devices board ยกเว้นใน assignment 8.5 ขา ACK ของ port DC3 ต้องเป็น input แผนผังอย่างคร่าวๆของ devices board



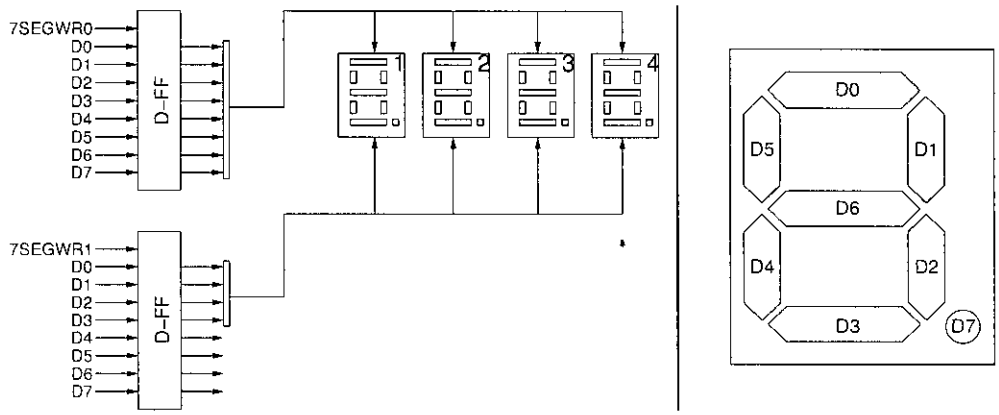
สัญญาณ RD, WR จะควบคุมให้ port J7 เป็น input หรือ output ถ้าจะอ่านค่าจาก devices board ให้ RD = 0 และ WR = 1, ถ้าจะเขียนค่าลง devices board ให้ RD = 1 และ WR = 0, A0 – A3 จะผ่าน Decoder เพื่อสร้างสัญญาณ read/write ไปควบคุมอุปกรณ์ต่างๆ ใช้ decoder 2 ตัว ตัวหนึ่งสร้างสัญญาณเพื่อ read devices อีกตัวสร้างสัญญาณเพื่อ write devices ดูรูป ในขณะใดขณะหนึ่ง จะอ่านหรือเขียน devices ได้เพียงตัวเดียว



สัญญาณ input ที่ใช้ decode จะใช้แค่ 3 ตัว คือ A2, A1, A0 เช่น ที่ decoder ตัวแรก ถ้า {A2, A1, A0} เท่ากับ 000 แล้ว DACWR จะเท่ากับ 0 และสัญญาณ output ขาอื่นๆ (1 – 7) จะเป็น 1 หมด สัญญาณ WR, RD, A3 ต่อเข้ากับขา enable ของ decoder ปกติ A3 จะเป็น 0 เสมอ หรือถ้าเราอยากให้ output ทุกขาของ decoder เป็น 1 หมด ก็ให้ A3 = 1

6.1 7-segments

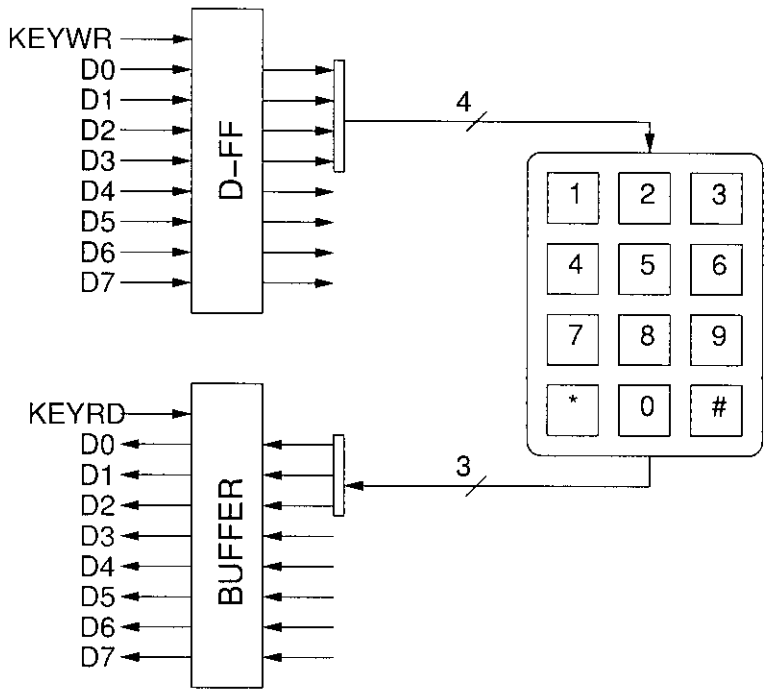
มี 7-segments ทั้งหมด 4 ตัว ตามวงจรที่ออกแบบไว้ 7-segments ทั้ง 4 ตัวจะแสดงค่าเหมือนกัน แต่เราเลือกได้ว่าจะให้ 7-segments อันไหนติดหรือดับ วิธีแสดงเลข 1, 2, 3, 4 ออกทาง 7-segments คือ แสดง 1 ที่ 7-segments อันแรก, แสดง 2 ที่ 7-segments อันที่สอง, แสดง 3 ที่ 7-segments อันที่สาม, แสดง 4 ที่ 7-segments อันที่สี่ และทำวนซ้ำเช่นนี้ไปเรื่อยๆด้วยความถี่พอประมาณ จะทำให้ตาคนเห็นว่า 7-segments แสดงค่า 1, 2, 3, 4 ออกมาพร้อมกัน มนุษย์ไม่สามารถมองเห็นความถี่ที่มากกว่า 50 Hz เช่น ถ้าไฟกะพริบเร็วกว่า 50 Hz เราจะเห็นว่ามันติดอยู่ตลอดเวลา ดังนั้นเราต้องทำให้ 7-segments แต่ละอันติดอยู่เป็นเวลา $\frac{1}{50} \times \frac{1}{4} = \frac{1}{200}$ วินาที (ต้องคูณ 1/4 เนื่องจากมี 7-segments 4 ตัว) ดูแผนผังของ 7-segments



D-FF จะ memory ค่า D0 – D7 ไว้เมื่อ 7SEGWR0 และ 7SEGWR1 เปลี่ยนจาก 0 เป็น 1 (positive edge) ในครั้งแรกเราจะ write ค่า ที่จะแสดงบน 7-segment ออกไปก่อนที่ D-FF ตัวแรก เช่น เลข 2 ต้องให้ {D0 – D7} = 11011010 ในการ write ครั้งที่สอง (D-FF ตัวล่าง) เราเลือกว่าจะให้ติดที่ 7-segments ตัวไหน โดยเซตค่า D3 – D0 ตามตารางนี้

7-segments no.	D3	D2	D1	D0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

6.2 Keyboard

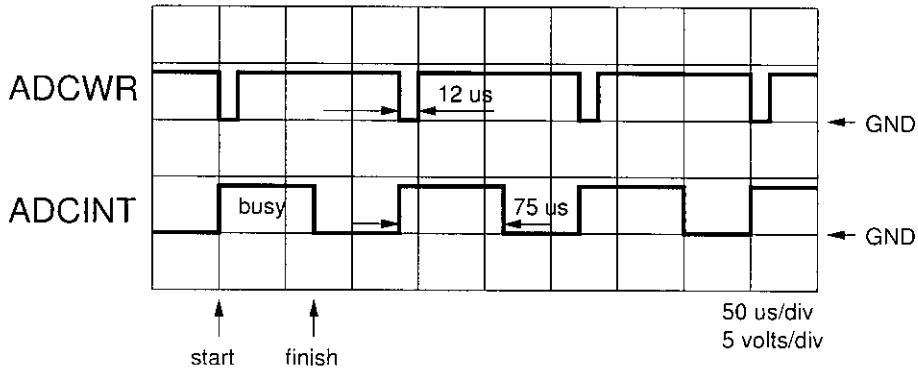


การตรวจสอบว่าปุ่มใดบน keyboard ถูกกด ต้องตรวจสอบทีละแถว เริ่มจากแถวแรก ให้ write {D3, D2, D1, D0} = 1110 (เปลี่ยนค่า KEYWR จาก 0 เป็น 1 เพื่อเขียนค่าลง D-FF) ต่อไป ให้อ่านค่า {D2, D1, D0} (ให้ KEYRD เป็น 0) ถ้าได้ 111 แสดงว่าไม่มีปุ่มใดในแถวนั้นถูกกด ถ้า {D2, D1, D0} = 110 หรือ 101 หรือ 011 แสดงว่าปุ่ม 1 หรือ 2 หรือ 3 ถูกกดตามลำดับ ต่อไปก็ตรวจสอบให้ครบทุกแถว โดย write {D3, D2, D1, D0} = 1101, 1011, 0111 เพื่อตรวจสอบแถวที่ 2, 3, 4 ตามลำดับ

6.3 Microphone

ปัญหาของการอ่านค่า (เสียง) จาก microphone คือ ต้องแปลงสัญญาณ analog เป็น digital ให้นิสิตศึกษาการทำงานของ A2D converter (ADC) จากเอกสาร http://161.200.93.31/

~u37cap/lab/a2d.pdf, ADC ทำงานด้วย clock ดังนั้นนิสิตต้องป้อน clock ให้กับ IC ตัวนี้ที่ขา ADCCLK ให้ใช้ ADCCLK = 1 MHz, ADC จะเริ่มแปลงสัญญาณ analog เป็น digital เมื่อเราให้ negative pulse ไป 1 ลูกที่ขา ADCWR เมื่อทำงานเสร็จ ADC จะส่งสัญญาณ interupt (negative edge) มาที่ขา ADCINT อย่าลืมเซต jumper ที่ FPGA board ให้ ADCINT เป็น input! (timing diagram ไม่ตรงกับใน datasheet เนื่องจากสัญญาณบน board ต้องผ่าน inverter)



ขอให้ใช้ pulse กว้างประมาณ 12 μs เพื่อกระตุ้นให้ ADC ทำงาน นิสิตต้องใช้ negative edge ของ ADCINT เป็น interupt เพื่อเตือนให้รู้ว่า ADC ทำงานเสร็จแล้ว เมื่อเกิด interupt ให้ขา ADCRD = 0 เพื่อ enable ค่าจาก ADC ลงมาที่ data bus (D0 – D7) ค่าที่ได้จะเป็น unsigned 8-bit integer, MSB อยู่ที่ D7 และ LSB อยู่ที่ D0 โดยปกติ นิสิตต้องเอาข้อมูลไปเก็บไว้ใน RAM, มีความต้านทานปรับค่าได้ชื่อ VR2 อยู่บน board ใช้สำหรับปรับความดังของเสียงเข้า

6.4 Speaker

นิสิตต้องใช้ digital-to-analog converter (DAC) เพื่อเปลี่ยนสัญญาณ digital เป็น analog วิธี write ข้อมูลลงไปบน DAC ก็เหมือน ICs ทั่วๆไปคือ เอาข้อมูลไปไว้บน data bus และทำให้ DACWR เป็น positive edge, มีความต้านทานปรับค่าได้ชื่อ VR3 อยู่บน board ใช้สำหรับปรับความดังของเสียงออก ดู datasheet ของ DAC ที่ http://161.200.93.31/~u37cap/lab/d2a.pdf

Elegance and truth are inversely related.

(Becker's Razor)

บทที่ 7

การตรวจสอบการทำงานของ FPGA board และ devices board

ในการทำ lab ที่ผ่านมา นิสิตมักจะอ้างว่าตัวเองเขียน Verilog ถูกแล้ว แต่ board เสีย ทำให้ได้ผลลัพธ์ผิด จากการตรวจสอบพบว่ามี FPGA board และ devices board ที่เสียจำนวนมาก (ซ่อมแล้ว) วิธีทดสอบคือ เขียน benchmark (Verilog) มา 8 ตัว เพื่อทดสอบกับ FPGA board และ devices board ถ้าได้ผลลัพธ์อย่างที่คาดไว้ ก็แสดงว่า board คงไม่เสีย นิสิตต้องเป็นผู้ตรวจสอบเอง benchmark ทั้งหมดให้ดาวโหลดจาก

http://161.200.93.31/~u37cap/lab/ (ไฟล์ 1.bit – 8.bit)

อย่างไรก็ตาม มักจะมีข้อผิดพลาดที่เกิดจากนิสิตเอง อาทิ เช่น power supply จ่ายไฟไม่พอ, Oscillator หรือ ICs ตัวอื่นๆเสียบไม่แน่น, เซต jumper ผิด, การต่อเชื่อมระหว่าง FPGA board กับ devices board ผิดหรือไม่แน่น, วงจรทำงานที่ความถี่สูงเกินไป, กำหนดขาของ FPGA ผิด ฯลฯ ถ้านิสิตไม่แน่ใจว่า board เสียหรือไม่ ให้ดาวโหลด benchmark มาทดสอบทุกครั้งที่ทำ lab

7.1 Benchmark 1: LEDs

ทำให้ LEDs ที่ DC0 กับ DC1 ติด/ดับ และ LEDs ที่ DC2 กับ DC3 ดับ/ติด สลับกันทุกๆ 1 วินาที ถ้ากดปุ่ม reset ค้างไว้ วงจรจะหยุดทำงาน

7.2 Benchmark 2: EEPROM

เมื่อกด reset วงจรจะวนอ่านค่าใน EEPROM ตั้งแต่ address แรก ถึง address สุดท้ายซ้ำกลับไปกลับมา (assume ว่าข้อมูลใน EEPROM คือ 0x55 0xaa . . . 0x55 0xaa) ถ้าค่าที่อ่านได้ไม่ถูกต้อง 8 บิทล่างของ DC2 จะเป็น 01010101 (ปกติจะเป็น 00000000) และหยุดการทำงาน

EEPROM มี 2^{15} address วงจรทำงานที่ความถี่ 1 MHz ดังนั้นใน 1 วินาที ก็อ่าน EEPROM ไปหลายรอบแล้ว ให้รอสักครู่ ถ้า 8 บิทล่างของ DC2 ไม่เป็น 01010101 ก็ถือ

ว่าวงจรทำงานถูก

7.3 Benchmark 3: RAM

คล้ายๆกับการทดสอบ EEPROM แต่ว่าเราต้องเขียนค่าลงไปใน RAM เอง ก่อนที่จะอ่านกลับมาตรวจสอบว่า ตรงกับค่าที่เราเขียนไปหรือไม่ ให้กด reset แล้วรอสักครู่ ถ้า 8 บิทล่างของ DC2 ไม่เป็น 01010101 ก็ถือว่าวงจรทำงานถูก

7.4 Benchmark 4: DC01

ต่อ port DC0 กับ DC2 และต่อ port DC1 กับ DC3 เซต jumper ให้ DC0, DC1 เป็น input และ DC2, DC3 เป็น output กด reset เพื่อเริ่มการทำงาน ถ้าวงจรทำงานถูกต้อง LEDs เริ่มติดจากดวงข้างล่างขึ้นไปข้างบน (ดวงที่ติดอยู่เดิมจะดับ) และกลับมาเริ่มข้างล่างต่อ

7.5 Benchmark 5: DC23

คล้ายกับ DC01 แค่เซต jumper ให้ DC0, DC1 เป็น output และ DC2, DC3 เป็น input

7.6 Benchmark 6: 7-segments

ต่อสายเชื่อมระหว่าง FPGA board กับ devices board ให้เรียบร้อย (อ่านบทที่ 6 Devices board ก่อน) เมื่อกด reset, 7-segments จะแสดงค่า "1234"

7.7 Benchmark 7: keyboard

เมื่อกด reset, 7-segments จะแสดงค่าที่กดบน keyboard

7.8 Benchmark 8: microphone/speaker

เมื่อกดปุ่ม reset, นิสิตมีเวลา 4 วินาทีที่จะบันทึกเสียงพูดผ่าน microphone เสียงจะถูกบันทึกใน RAM เพื่อนำมาเล่นกลับผ่านลำโพง โดยเล่นซ้ำทุกๆ 4 วินาที

For every complex problem,

there is a solution that is simple, neat, and wrong.

(H. L. Mencken)

บทที่ 8

Assignments and Term Project

8.1 Counter and dice (สัปดาห์ที่ 1, 5 คะแนน)

(1) สร้างวงจรหารความถี่จาก 8 MHz เป็น 1 Hz (2) สร้างวงจร binary counter ขนาด 4 บิทที่นับทีละ 1 วินาที และแสดงผลออกมาที่ LEDs (0 ให้ LEDs ดับ, 1 ให้ LEDs ติด) กำหนดให้ปุ่ม reset เป็นการเคลียร์ counter (3) สร้างวงจรลูกเต๋า เมื่อกดปุ่ม reset จะให้ค่า random ระหว่าง 1 – 6 ออกมาที่ LEDs

8.2 EEPROM and RAM (สัปดาห์ที่ 2, 5 คะแนน)

(1) สร้างวงจรที่นำค่าที่อยู่ใน EEPROM address 0000 – 000F มาคำนวณหา check-sum ขนาด 8 บิท และแสดงผลลัพธ์ออกมาที่ LEDs (0 ให้ LEDs ดับ, 1 ให้ LEDs ติด) (2) สร้างวงจรที่นำค่าที่อยู่ใน EEPROM address 0000 – 000F ไปเก็บไว้ใน memory ที่ address เดียวกัน และแสดงค่าใน RAM ออกมาที่ LEDs ทุก ๆครึ่งวินาทีตามลำดับจาก address 0000 – 000F (นิสิตต้องฝึกใช้ RAM เพราะต้องใช้ใน assignment ต่อไปและใน term project)

8.3 7-segments (สัปดาห์ที่ 3, 5 คะแนน)

ให้แสดงเลขประจำตัวนิสิต (เช่น 4371804321) ที่ 7-segments ทีละ 4 digit และ rotate left ไปทีละ 1 digit ทุก 1 วินาที ถ้ากด reset ให้กลับมาเริ่มที่ digit แรก (ถ้ามองเห็นว่า 7-segments กะพริบจะไม่ได้คะแนน)

8.4 Keyboard (สัปดาห์ที่ 4, 10 คะแนน)

สร้างวงจรที่ เมื่อกดปุ่ม reset ให้เริ่มทำงานดังนี้ initialize stack, เมื่อกด keyboard (0 – 9 และ *) วงจรจะเก็บค่าที่เรากดไว้ใน stack เมื่อกด # จะ pop ของใน stack, 7-segments ตัวขวาสุด จะแสดงค่า top-of-stack ถ้า stack ว่างเปล่าให้แสดง E บน 7-

segments (* ให้วาด 7-segments เป็นตัว A, และ ถ้ามองเห็นว่า 7-segments กะพริบจะไม่ได้คะแนน)

8.5 Microphone and speaker (สัปดาห์ที่ 5-6, 15 คะแนน)

สร้างวงจรที่ เมื่อกด reset จะแปลงสัญญาณเสียงเป็น digital และเก็บไว้ใน RAM กำหนด sampling rate = 8 kHz (แปลงสัญญาณ analog เป็น digital ทุก ๆ $\frac{1}{8,000}$ วินาที) เมื่อเขียน RAM ถึง address สุดท้าย ให้เล่นเสียงกลับทางลำโพง (เขียนข้อมูลไปที่ DAC ทุก ๆ $\frac{1}{8,000}$ วินาที) เมื่อเล่นจบให้เล่นซ้ำไปเรื่อย ๆ (ต้องให้ได้เสียงพูดกลับมาเหมือนเดิม เสียงพูดต้องไม่เร็วหรือช้ากว่าเดิม ห้ามมีเสียงซ่ารบกวน)

8.6 Midterm Exam (สัปดาห์ที่ 7, 20 คะแนน)

หลังจาก assignment สุดท้าย นิสิตต้องสอบ midterm สอบเดี่ยว (ไม่สอบเป็นกลุ่ม) ให้เวลาประมาณ 1 ชม. ข้อสอบจะไม่ยากมาก (ประมาณ FSM ที่มีไม่เกิน 4 states)

8.7 Term project (สัปดาห์ที่ 8 ถึงสัปดาห์สุดท้าย, 40 คะแนน)

ข้อมูลเสียงเพลง (sampling rate 8 kHz) ถูกบีบอัดข้อมูลและเข้ารหัสอยู่ใน EEP-ROM 4 ตัว (4 เพลง) ให้ออกแบบวงจรถอดรหัสและคลายข้อมูล บอกชื่อเพลงถูก 1 เพลงได้ 10 คะแนน อัลกอริทึมที่ใช้ถอดรหัสและคลายข้อมูล ให้ดาวโหลดจาก http://161.200.93.31/~u37cap/lab/project.pdf