CPU Simulation

yen3
SoC Lab, CGUCSIE

January 7, 2010

1 Instrodution

這份文件寫的相當的匆促，如果有任何問題及錯誤，歡迎與助教或老師討論。

在 Computer Organization Lab Final Project 中，我們要模擬出一顆 CPU，只使用模擬而不燒到 FPGA 版上的原因是，Instruction Memory 與 Data Memory 是較難以燒錄到 FPGA 版上順利運行的，所以我們採用另外一個方式。

2 Spec

Define Data Width and Memory Size

- Register – $2^3 \times 8$ bits, each register has 8 bits
- Instruction Memory — $2^8 \times 16$ bits, each memory address has 16 bits
- Data Memory — $2^8 \times 8$ bits, each memory address has 8 bits

Define Memory Interface

- Instruction Memory Interface

```
wire [7:0] i_addr; // Instruction Memory Address
wire [15:0] instr; // instruction in instr_memory[i_addr]
```

- Data Memory Interface

```
wire [7:0] d_addr; // memory address
wire [7:0] d_indata; // input data
wire mw; // memory write enable, if(mw == 1'b1) data_memory[d_addr] <= d_indata;
wire [7:0] d_outdata; // output data, d_outdata = data_memory[d_addr]
```

3 Start

3.1 CPU Module Interface

首先，我們的期末 project 到底要做什麼？ 很顯然是做一顆 CPU，那麼我們應該要做到什麼，在課本的圖中，應該除了 Instruction Memory 和 Data Memory 的 block 都要完成，寫成一個很像下面的 module

CPU Interface

```
module cpu(
    input clean,
```

```
    input clk,
    input [15:0] instr,

    input [7:0] d_datain,
    output d_mw,
    output [7:0] d_address, // d_addr
    output [7:0] d_dataout,

    output [7:0] pc // i_addr
);
```

其實這個 cpu module 完成了除了 IM 與 DM 的其於部分，只留下了接 IM 與 DM 的 input/output，到這一步之前，都有辦法在 Quartus II 上完成模擬，所以要連接上 IM 與 DM 時，先確保除了 IM 與 DM 以外的功能皆是正確的，而且確保至少有這些 input/ output，那麼我們就能進行下一步。

3.2 Link to testbench_cpu.v

假設你已經做好上個步驟了，那麼接下來我們就準備連接了，從 SoC Lab 課程 wiki 上下載下來 testbench_cpu.v 打開之後會長成這樣。

testbench_cpu.v

```
module main;

reg clk;
reg [31:0] sim_cycle;

initial clk = 0;
initial sim_cycle = 0;

// Set clock and sim_cycle
always #5 clk = ~clk;
always @(posedge clk) sim_cycle <= sim_cycle + 1;


// Instruction Memory
wire [7:0] i_addr;
wire [15:0] instr;

reg [15:0] instr_memory[0:255];
initial begin
    $readmemb("instr.mem", instr_memory);
end
assign instr = instr_memory[i_addr];


// Data Memory
wire mw;
wire [7:0] d_addr;
wire [7:0] d_indata;
wire [7:0] d_outdata;

reg [7:0] data_memory[0:255];
initial begin
    $readmemh("data.mem", data_memory);
end
```

```
35
36  always @(posedge clk) begin
37      if(mw == 1'b1) begin
38          data_memory[d_addr] <= d_indata;
39      end
40  end
41  assign d_outdata = data_memory[d_addr];
42
43  // CPU Module Linking Example
44  // You need to modify the block for link to your module.
45  reg clean;
46  cpu cpu8(
47      .clean(clean),
48      .clk(clk),
49      .instr(instr),
50      .dm_mw(mw),
51      .dm_datain(d_outdata),
52      .dm_dataout(d_indata),
53      .dm_address(d_addr),
54      .pc(i_addr)
55  );
56
57
58  //to feed test vectors
59  always @(posedge clk) begin
60      case (sim_cycle)
61          0: begin clean = 1'b1; end
62          default: begin clean = 1'b0; end
63      endcase
64  end
65
66  initial begin
67          while (sim_cycle<256) begin
68                  @(posedge clk);
69                  #2;
70          if(instr == 16'hffff)
71              $finish(0);
72
73          $display("%d: run", sim_cycle);
74      end
75
76          $finish(0);
77  end
78
79  endmodule
```

而在 46 到 54 行，這邊只是一個示範，也就是說，你必需將這個 CPU module 改成你自己的 module，就可以完成連接了。如果對整隻程式有興趣，不妨自行研究，有問題歡迎討論。

在完成連接之後，如何測試是否有連接成功或編譯錯誤的話，就要使用 ModelSim 來 compile 這所有的檔案，而我在另外一份簡報 “ModelSim Simple Example” 附有簡單的使用說明及範例。

4 How to produce Data

4.1 How to produce Instruction Memory

在 testbench_cpu.v 中的 20 行是

testbench_cpu.v

```
$readmemb("instr.mem", instr_memory);
```

這一行的意思是，我們會從 instr.mem 中讀取資料，接著塞到 Instruction Memory 中，也就是說，我們在 ModelSim 模擬的時候，在專案資料下放入 instr.mem (不能是別的檔名) 就可以了。

那麼這個檔案的格式會長成如下，每一行代表一個記憶體。

instr.mem

```
0000000100101000
0000001101110000
0000010001010011
0000101011100101
0000110110111000
0001000001001011
0001001010011100
0001010010011100
0001011101110000
0001100101000110
...
```

其實你可以用手慢慢打入每一行指令，最多打 240 個指令，或者是用一個比較好一點的方法是，助教與老師在課堂之餘用 Python 寫了一個 assembler 轉譯組合語言的程式碼。

假設 Python 已經裝好也會設定了(這在另外一份簡報會描述)，我們寫了一個 instr.asm 如下

instr.asm

```
MOVA R4, R5
INC R5, R6
ADD R1, R2, R3
SUB R3, R4, R5
DEC R6, R7
AND R1, R1, R3
OR R2, R3, R4
XOR R2, R3, R4
NOT R5, R6
MOVB R5, R6
```

打開 command line 執行

Command line

```
python ass.py instr.asm
```

就可以很順利的自動幫你產生一個 .mem file，讓你使用，將產生的 .mem file 改名成 instr.mem 之後放入 ModelSim 執行的專案資料下即可。

4.2 How to produce Data Memory

在 testbench_cpu.v 中的 33 行是

testbench_cpu.v

```
$readmemh("data.mem", data_memory);
```

意思與 Instruction Memory 的讀取是一樣的，唯一不一樣的是檔案格式不同。檔案格式是一個 hex format 會長成

instr.mem

```
0c
22
ff
00
01
02
03
04
38
00
```

一樣，我們寫了一個很簡單的 Python 小程式，假設我們寫了一個這樣子的檔案如下

data.data

```
12
34
255
0
1
2
3
4
56
0
```

打開 command line，輸入如下的指令

Command line

```
python data_mem.py data.data
```

就會生一個 .mem file ，將產生的 .mem file 改名成 data.mem 之後放入 ModelSim 執行的專案資料下即可。

5 Simulation

根據 “ModelSim Simple Example” 簡報，在專案資料夾中放入所有的程式檔案 (*.v) 及 instr.mem 與 data.mem ，應該就可以順利執行了。

還是再說一次，如果有任何問題，歡迎直接與助教或老師討論，雖然我們也不太會這些軟體就是了 XD。