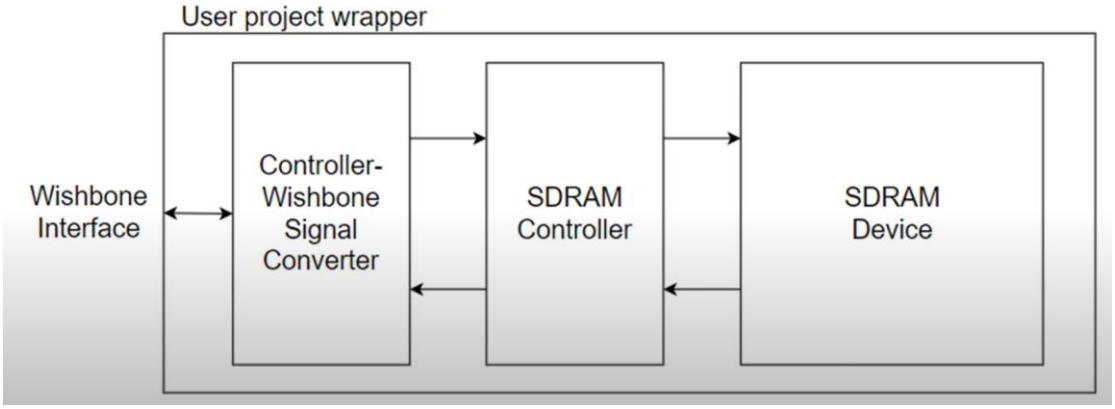
NYCU 電子研究所 系統晶片設計 LAB D

呂紹愷 311510187 江尹凡 311510207 廖智緯 311510216

- Introduction

  此次 Lab 與 4-1 十分相似，只是需要把 4-1 中的 BRAM 換成 SDRAM+SDRAM-controller，如下圖所示。

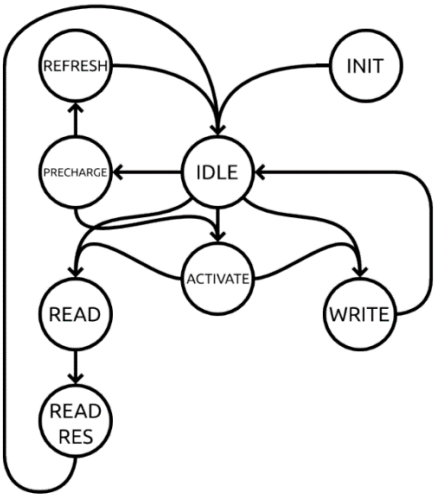


  由於 SDRAM-controller 無法直接與 wishbone 做溝通，因此中間需要再透過一個 converter 來做轉換。

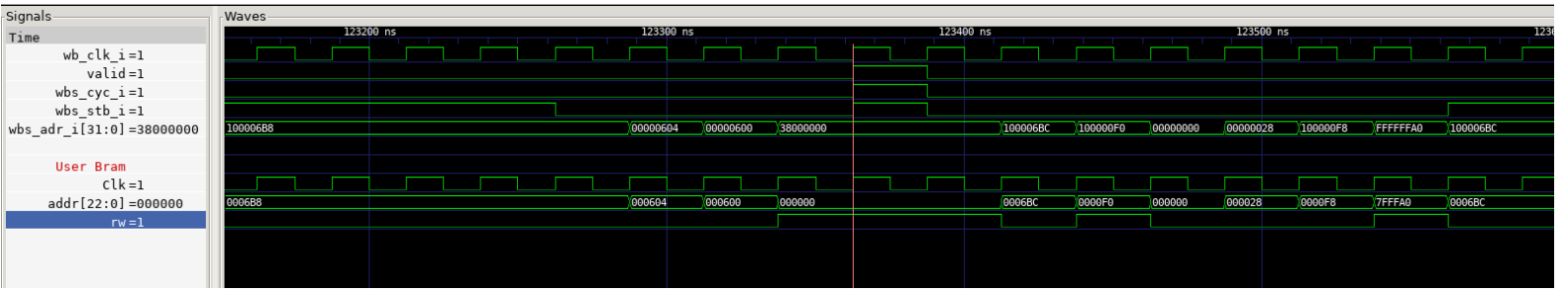
  - SDRAM Device
    - sdram_cle, sdram_cs, sdram_cas, sdram_ras, sdram_we
    - sdram_dqm, sdram_ba, sdram_a
    - sdram_dqi, sdram_dqo
  - User interface
    - user_addr
    - rw
    - data_in, data_out
    - busy
    - in_valid, out_valid

SDRAM Controller

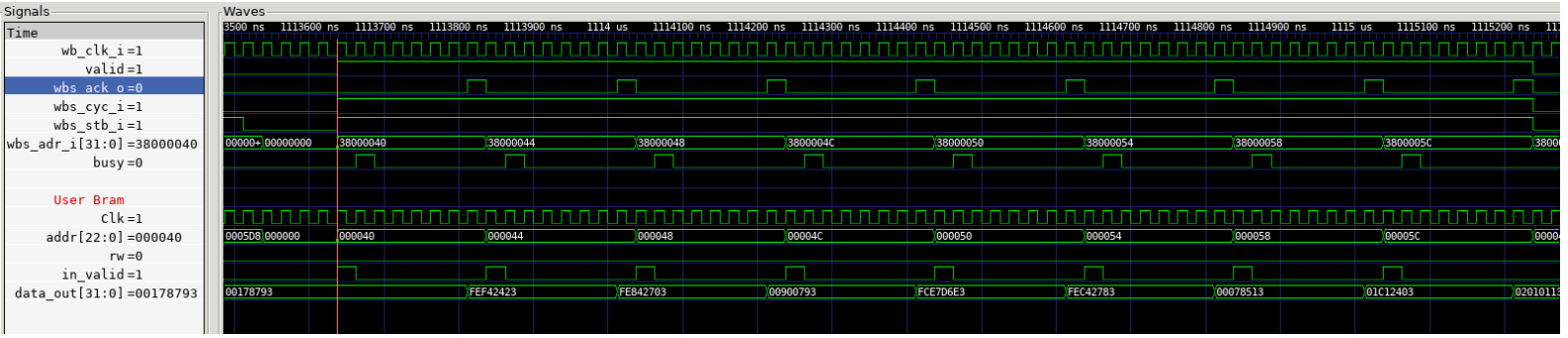
- INIT→IDLE
- IDLE→ACTIVATE→WRITE→IDLE
- IDLE→ACTIVATE→READ→READ_RES →IDLE
- IDLE→WRITE→IDLE
- IDLE→READ→READ_RES→IDLE
- IDLE→PRECHARE→ACTIVATE→WRITE →IDLE
- IDLE→PRECHARE→ACTIVATE→READ →READ_RES→IDLE
- IDLE→PRECHARE→REFRESH→IDLE

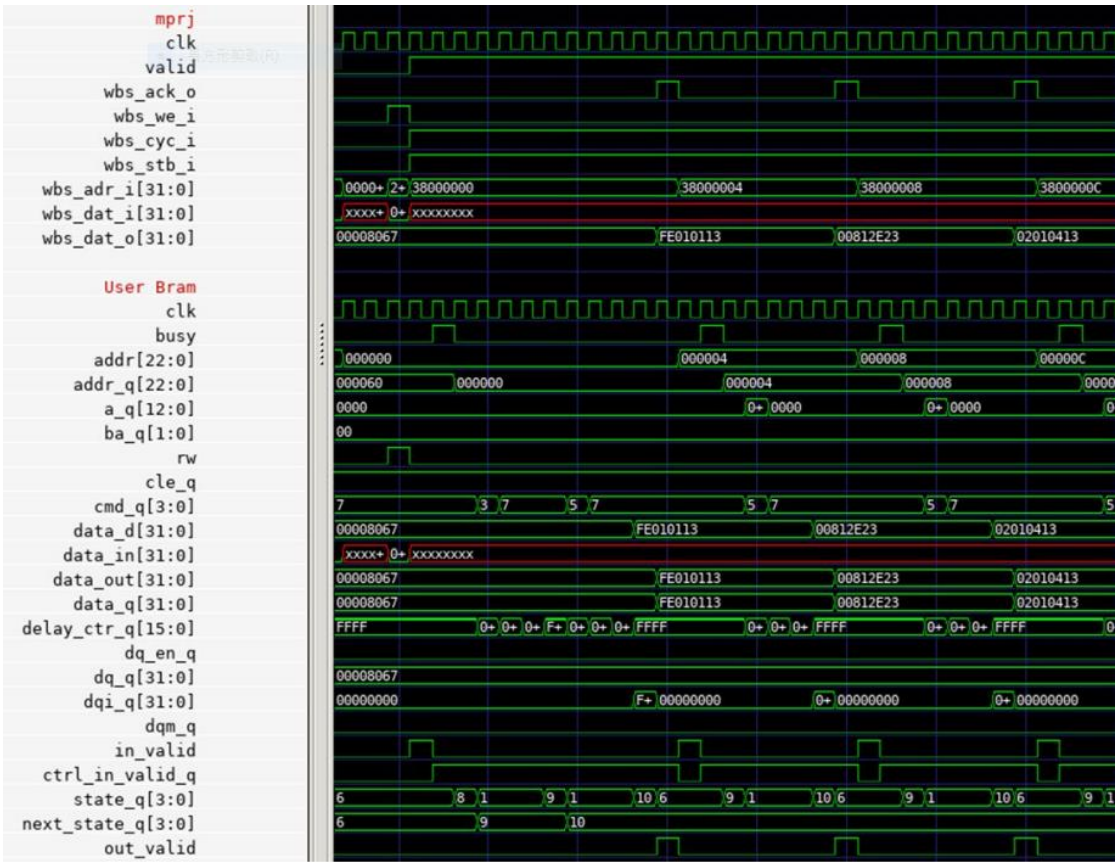


write



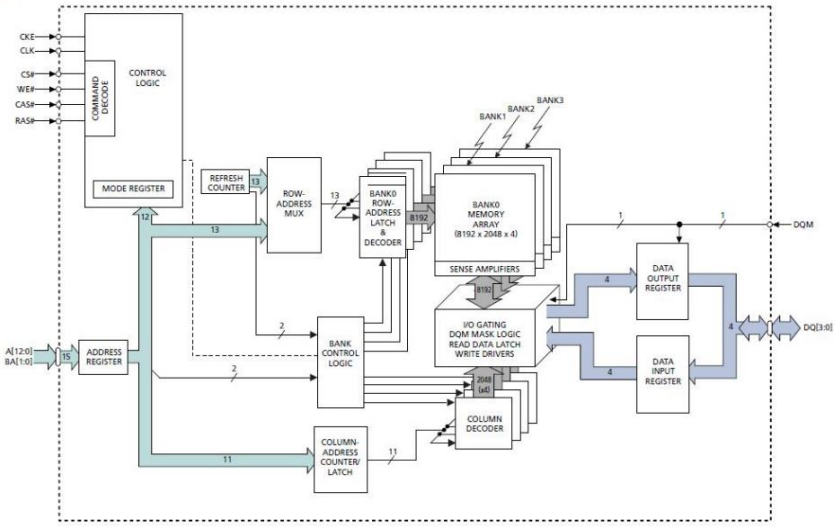
read (連續讀取) 經觀察可發現，本實驗中會連續讀取 8 筆 code 資訊，且地址為連續。





我們可以觀察到，每次讀取都會跑回 idle(6)，這浪費了 3 個 cycle 的時間，而若有 prefetch 則可大幅降低時間。

- The behavior model refer to Micron MT48LC64M4A2
  - 16 Meg x 4 x 4 banks



- Prefetch

我們 prefetch 的方式是去猜測下一筆地址為 addr+4，不等 wishbond 收到 outvalid ，我們就提早先把+4 的地址丟給 sdram controller，這樣做可以減少 sdram controller 在 idle(6)等的時間。prefetch 的判斷方式為現在 sdram 已經在讀 data 了，而 wishbond 還有持續 request

```
assign user_addr = (prefetch && valid && !wbs_we_i ) ? next_addr : ctrl_addr;

always@(posedge clk)begin
    if(rst)
        next_addr <= 0;
    else
        next_addr <= (ctrl_in_valid && !wbs_we_i) ? user_addr + 4 : next_addr;
end

always@(posedge clk)begin
    if(rst)
        prefetch <= 0;
    else begin
        if(ctrl_out_valid && !wbs_we_i &&prefetch)
            prefetch <= (now_in_addr ==  wbs_adr_i[22:0])? prefetch:0;
        else if(valid && !wbs_we_i)
            prefetch <= (ctrl_in_valid_q) ? 1 : prefetch;
        else
            prefetch <= 0;
    end
end
```

由於我們是去猜測地址，怕 prefetch 的地址猜錯 (不是+4) ，所以在 wbs_ack_o 要去判斷是不是猜對 addr 猜錯這筆從 sdram 讀出來的就不能輸出。

```
assign valid = wbs_stb_i && wbs_cyc_i;
assign ctrl_in_valid = wbs_we_i ? valid : ~ctrl_in_valid_q && valid && !ctrl_busy;
assign wbs_ack_o = (wbs_we_i) ? ~ctrl_busy && valid : ctrl_out_valid && valid &&(now_in_addr ==  wbs_adr_i[22:0]) ;
assign ctrl_addr = wbs_adr_i[22:0];
```

Ctrl_out_valid 也要改成一但不 busy 就拉 0(才可以輸入下一筆 data)

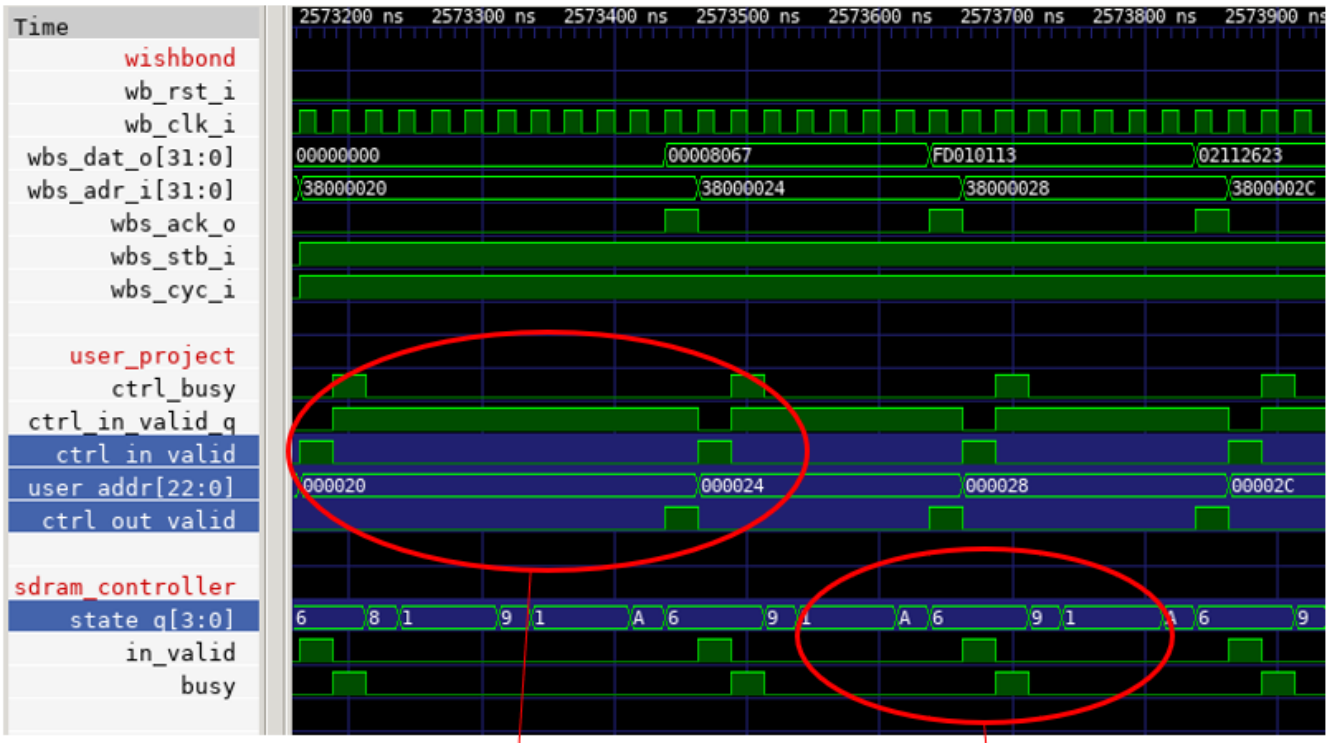
```
always @(posedge clk) begin
    if (rst) begin
        ctrl_in_valid_q <= 1'b0;
    end
    else begin
        if (~wbs_we_i && valid && ~ctrl_busy && ctrl_in_valid_q == 1'b0)
            ctrl_in_valid_q <= 1'b1;
        else if (ctrl_out_valid || ~ctrl_busy)
            ctrl_in_valid_q <= 1'b0;
    end
end
```

要輸出時要確認地址是否正確(可能猜錯不是+4)
猜錯要關掉prefetch，去讀取對的addr

第一筆sdram還沒讀出來
就先invald輸入第二筆的addr (猜測為addr+4)
圖中圈選位置可以看到第1筆data還沒讀出來我們就先輸入下一筆的addr

sdram就不會卡在IDLE(6)3T

原本的 SRAM



等到ctrl_out_valid(sdram讀出data)，ctrl_in_valid_q才歸0(歸0才可以輸入下一筆addr)，表示都是要等到前一筆的outvalid、wbs_ack_o回傳會去，才會輸入下一個data的addr

Sdram會卡在IDLE(6)3T

- Introduce the bank interleave for code and data
  在 section.lds 中可以修改 code 和 data 的地址，要去調整 bank 可以修改 addr[9:8]，我們將 code 放到 0x38000000~0x38000200，對應至 bank0, 1；data 放到 0x38000200~0x38000400，對應至 bank2, 3。

```
MEMORY {
    vexriscv_debug : ORIGIN = 0xf00f0000, LENGTH = 0x00000100
    dff : ORIGIN = 0x00000000, LENGTH = 0x00000400
    dff2 : ORIGIN = 0x00000400, LENGTH = 0x00000200
    flash : ORIGIN = 0x10000000, LENGTH = 0x01000000
    mprj : ORIGIN = 0x30000000, LENGTH = 0x00100000
    SRAM_code : ORIGIN = 0x38000000, LENGTH = 0x00000200
    SRAM_data : ORIGIN = 0x38000200, LENGTH = 0x00000200
    hk : ORIGIN = 0x26000000, LENGTH = 0x00100000
    csr : ORIGIN = 0xf0000000, LENGTH = 0x00010000
}
```

```
    .data :
    {
        . = ALIGN(8);
        _fdata = .;
        *(.data .data.* .gnu.linkonce.d.*)
        *(.data1)
        _gp = ALIGN(16);
        *(.sdata .sdata.* .gnu.linkonce.s.*)
        . = ALIGN(8);
        _edata = .;
/*  } > dff AT > flash */
    } > SRAM_data AT > flash

    .bss :
    {
        . = ALIGN(8);
        _fbss = .;
        *(.dynsbss)
        *(.sbss .sbss.* .gnu.linkonce.sb.*)
        *(.scommon)
        *(.dynbss)
        *(.bss .bss.* .gnu.linkonce.b.*)
        *(COMMON)
        . = ALIGN(8);
        _ebss = .;
        _end = .;
/*  } > dff AT > flash */
    } > SRAM_data AT > flash

    .mprjram :
    {
        . = ALIGN(8);
        _fsram = .;
        *libgcc.a:*(.text .text.*)
    } > SRAM_code AT > flash
}
```