

112-1 Soc Design Laboratory

Lab D SDRAM

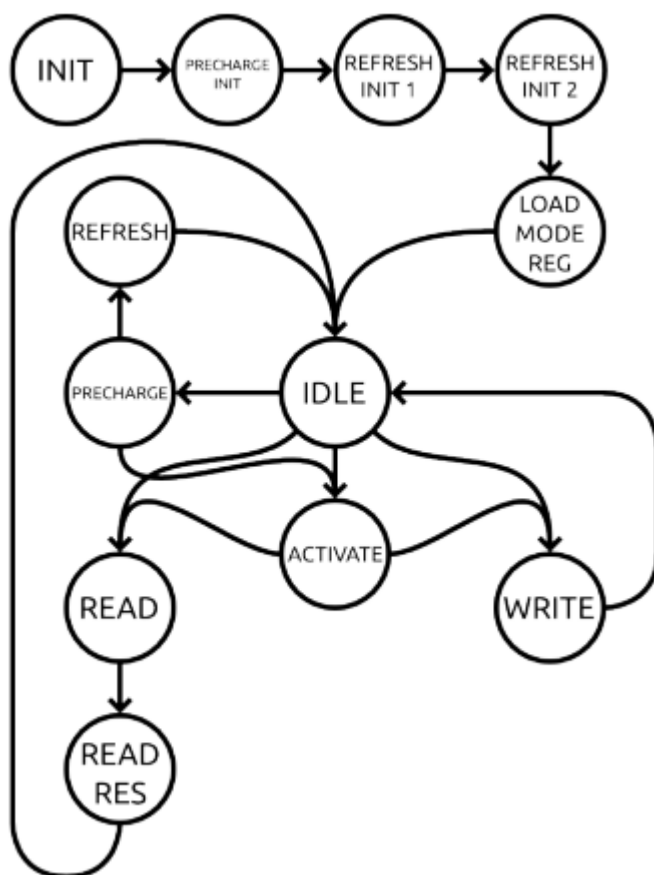
組員：丁續翰、潘金生、王證皓

1. SDRAM Controller

(1)FSM

如下圖 FSM 所示，INIT 狀態主要將所有值做初始化，

IDLE 狀態會先偵測是否需 refresh，如不用的話，接著會偵測 row 是否有 open，有的話會選擇執行 READ 以及 WRITE 或跳至 PRECHARGE，無的話則會執行 ACTIVATE。



(2)控制訊號

在主要控制訊號的部分，會有 user_addr、rw、data_in、

data_out、busy、in_valid、out_valid，其中 user_addr 表示

s dram 的位置，rw 為 1 則為寫，0 為讀，data_in 與 data_out 為 data 的輸入輸出，並由 in_valid 以及 out_valid 來代表有效的資料，busy 表示 controller 目前的狀態是否能接收下一個命令。

(3)Bus Protocol

```
// WB MI A
assign valid = wbs_stb_i && wbs_cyc_i;
assign ctrl_in_valid = wbs_we_i ? valid : ~ctrl_in_valid_q && valid;
assign wbs_ack_o = (wbs_we_i) ? ~ctrl_busy && valid : ctrl_out_valid;
assign bram_mask = wbs_sel_i & {4{wbs_we_i}};
assign ctrl_addr = wbs_adr_i[22:0];

// IO
assign io_out = d2c_data;
assign io_oeb = {('MPRJ_IO_PADS-1){rst}};

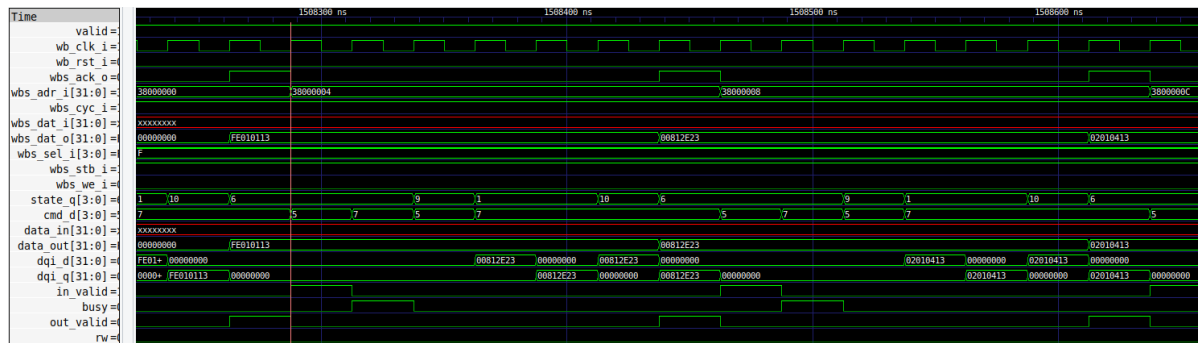
// IRQ
assign irq = 3'b000; // Unused

// LA
assign la_data_out = {{{(127-BITS){1'b0}}}, d2c_data};
// Assuming LA probes [65:64] are for controlling the count clk & reset
assign clk = (~la_oenb[64]) ? la_data_in[64]: wb_clk_i;
assign rst = (~la_oenb[65]) ? la_data_in[65]: wb_rst_i;
assign rst_n = ~rst;

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_in_valid_q <= 1'b0;
    end
end
```

因 wishbone 只有一個 wbs_ack_o，所以須透過 converter 將訊號做轉換，當要寫入 s dram 時，需先判斷 wbs_we_i 是否為 1，並判斷狀態不能為 busy 與 valid，而要讀 s dram 時，則只需觀察 out_valid 及代表讀出資料有效。

2. Prefetch Scheme



上圖波形圖為使用 prefetch 後的結果，實現方法為在 IDLE

狀態(state_q=6)以及 busy 為 0 時，預先發出

READ(cmd_d=5)的 command 來 prefetch 資料，如此就可將

資料預先讀好，並在 READ 狀態時，能更快將資料讀出，

原本 READ 狀態後須等待 3 個 T，但使用 prefetch 後，只需

等待兩個 T。

3. Bak Interleave

```
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
    mprjram : ORIGIN = 0x38000000, LENGTH = 0x00400000
    all_data : ORIGIN = 0x38000200, LENGTH = 0x00000600
    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 = .;
} > all_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 = .;
} > all_data AT > flash
```

```

38000228 <A>:
38000228:      0000      .2byte 0x0
3800022a:      0000      .2byte 0x0
3800022c:      0001      .2byte 0x1
3800022e:      0000      .2byte 0x0
38000230:      0002      .2byte 0x2
38000232:      0000      .2byte 0x0
38000234:      00000003   lb      zero,0(zero) # 0 <__DYNAMIC>
38000238:      0000      .2byte 0x0
3800023a:      0000      .2byte 0x0
3800023c:      0001      .2byte 0x1
3800023e:      0000      .2byte 0x0
38000240:      0002      .2byte 0x2
38000242:      0000      .2byte 0x0
38000244:      00000003   lb      zero,0(zero) # 0 <__DYNAMIC>
38000248:      0000      .2byte 0x0
3800024a:      0000      .2byte 0x0
3800024c:      0001      .2byte 0x1
3800024e:      0000      .2byte 0x0
38000250:      0002      .2byte 0x2
38000252:      0000      .2byte 0x0
38000254:      00000003   lb      zero,0(zero) # 0 <__DYNAMIC>
38000258:      0000      .2byte 0x0
3800025a:      0000      .2byte 0x0
3800025c:      0001      .2byte 0x1
3800025e:      0000      .2byte 0x0
38000260:      0002      .2byte 0x2
38000262:      0000      .2byte 0x0
38000264:      00000003   lb      zero,0(zero) # 0 <__DYNAMIC>

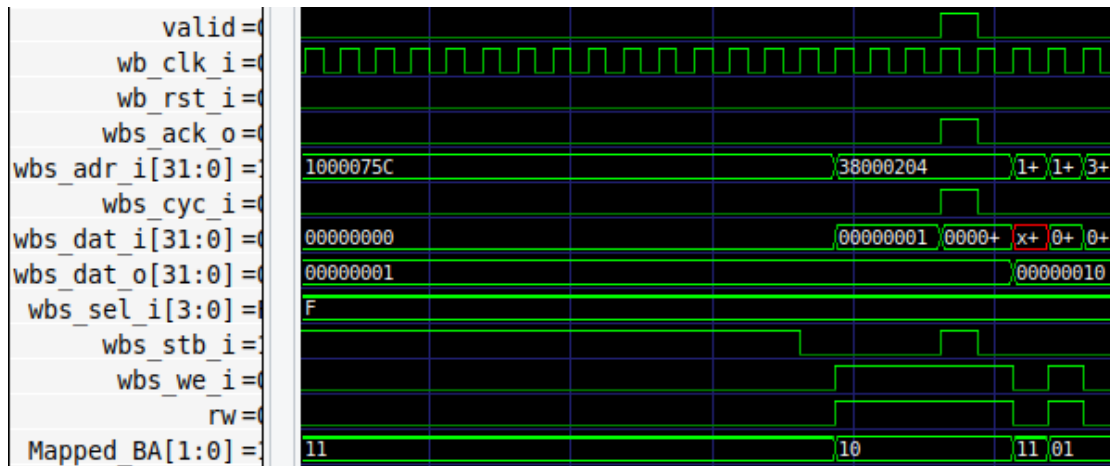
38000268 <B>:
38000268:      0001      .2byte 0x1
3800026a:      0000      .2byte 0x0
3800026c:      0002      .2byte 0x2
3800026e:      0000      .2byte 0x0
38000270:      00000003   lb      zero,0(zero) # 0 <__DYNAMIC>
38000274:      0004      .2byte 0x4
38000276:      0000      .2byte 0x0
38000278:      0005      .2byte 0x5
3800027a:      0000      .2byte 0x0
3800027c:      0006      .2byte 0x6
3800027e:      0000      .2byte 0x0
38000280:      00000007   .4byte 0x7
38000284:      0008      .2byte 0x8
38000286:      0000      .2byte 0x0

```

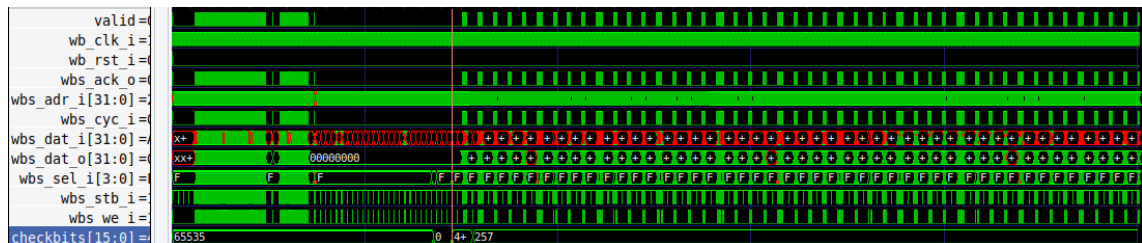
如上圖，透過修改 linker，將 data 與 code 放在不同的

bank，matmul 的 data(A、B)皆會放在 2 號 bank 裡，如波形

所示，bank address 為 2。



4. Access Conflicts



如上圖所示，降低 refresh period 至 100 的 cycle 時，sdrām 無法及時完成 refresh 的動作，因而導致讀寫的資料產生錯誤，無法正常運作。