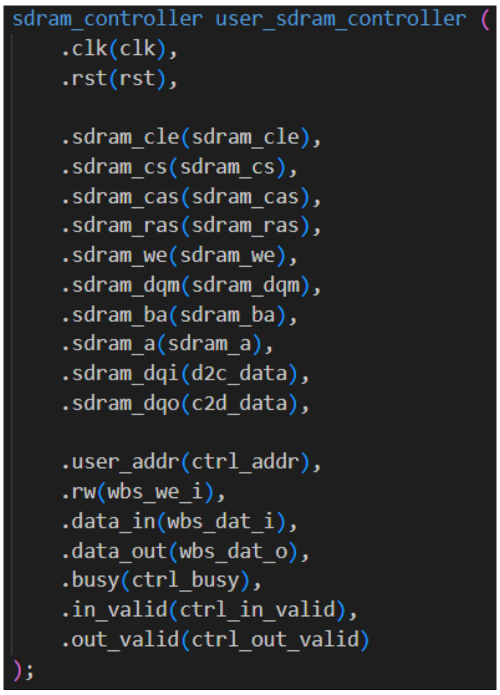
Lab-D Report

組別 : 第1組

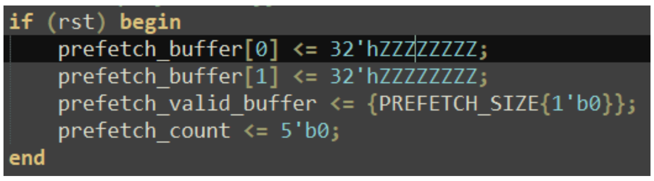
組員 : 游承緯 312580044   符顥瀚 312510154

- SDRAM controller design
  - Interface



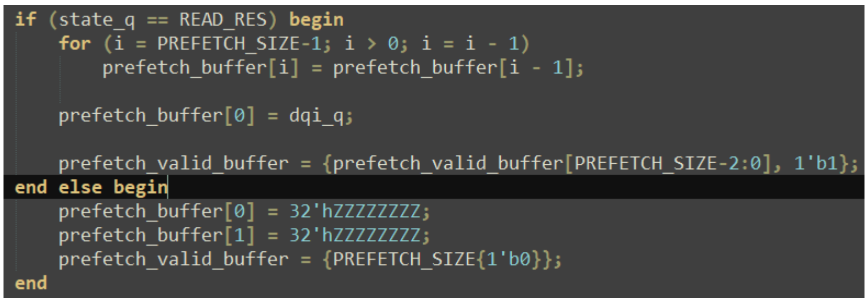
```
sdram_controller user_sdram_controller (
    .clk(clk),
    .rst(rst),

    .sdram_cle(sdram_cle),
    .sdram_cs(sdram_cs),
    .sdram_cas(sdram_cas),
    .sdram_ras(sdram_ras),
    .sdram_we(sdram_we),
    .sdram_dqm(sdram_dqm),
    .sdram_ba(sdram_ba),
    .sdram_a(sdram_a),
    .sdram_dqi(d2c_data),
    .sdram_dqo(c2d_data),

    .user_addr(ctrl_addr),
    .rw(wbs_we_i),
    .data_in(wbs_dat_i),
    .data_out(wbs_dat_o),
    .busy(ctrl_busy),
    .in_valid(ctrl_in_valid),
    .out_valid(ctrl_out_valid)
);
```

  - Reset prefetch buffer



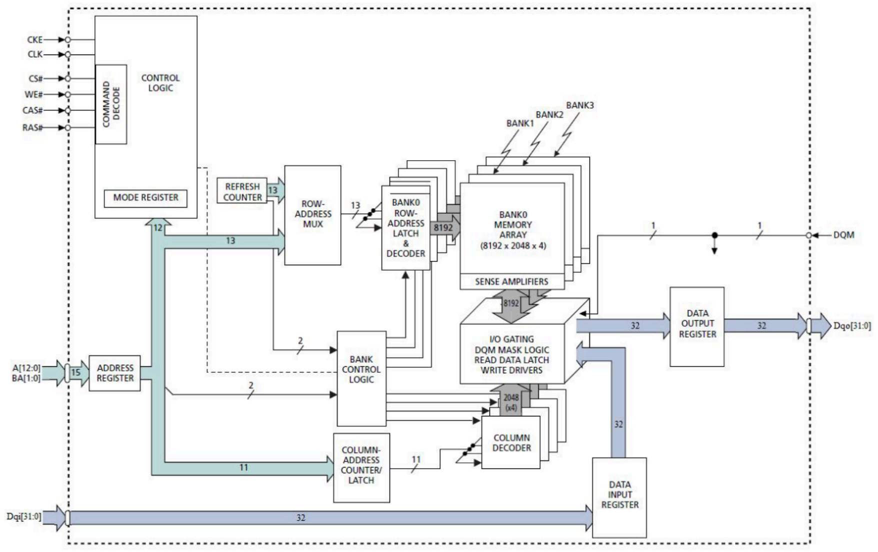
```
if (rst) begin
    prefetch_buffer[0] <= 32'hZZZZZZZZ;
    prefetch_buffer[1] <= 32'hZZZZZZZZ;
    prefetch_valid_buffer <= {PREFETCH_SIZE{1'b0}};
    prefetch_count <= 5'b0;
end
```

  - When read request

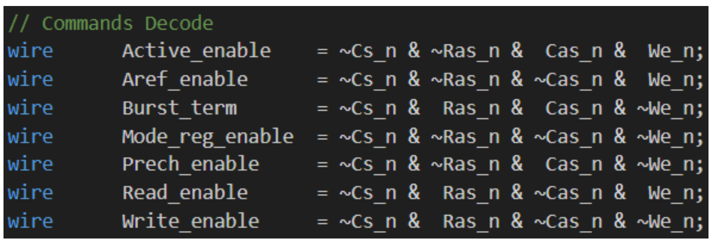


```
if (state_q == READ_RES) begin
    for (i = PREFETCH_SIZE-1; i > 0; i = i - 1)
        prefetch_buffer[i] = prefetch_buffer[i - 1];

    prefetch_buffer[0] = dqi_q;

    prefetch_valid_buffer = {prefetch_valid_buffer[PREFETCH_SIZE-2:0], 1'b1};
end else begin
    prefetch_buffer[0] = 32'hZZZZZZZZ;
    prefetch_buffer[1] = 32'hZZZZZZZZ;
    prefetch_valid_buffer = {PREFETCH_SIZE{1'b0}};
end
```

- SDRAM bus protocol
  - block diagram



  - decode



```
// Commands Decode
wire      Active_enable     = ~Cs_n & ~Ras_n &  Cas_n &  We_n;
wire      Aref_enable       = ~Cs_n & ~Ras_n & ~Cas_n &  We_n;
wire      Burst_term        = ~Cs_n &  Ras_n &  Cas_n & ~We_n;
wire      Mode_reg_enable   = ~Cs_n & ~Ras_n & ~Cas_n & ~We_n;
wire      Prech_enable      = ~Cs_n & ~Ras_n &  Cas_n & ~We_n;
wire      Read_enable       = ~Cs_n &  Ras_n & ~Cas_n &  We_n;
wire      Write_enable      = ~Cs_n &  Ras_n & ~Cas_n & ~We_n;
```

將SDRAM的四個訊號(CS、RAS、CAS、WE) decode 為7個狀態，各個狀態需要執行的描述則會在sdr.v裡面執行

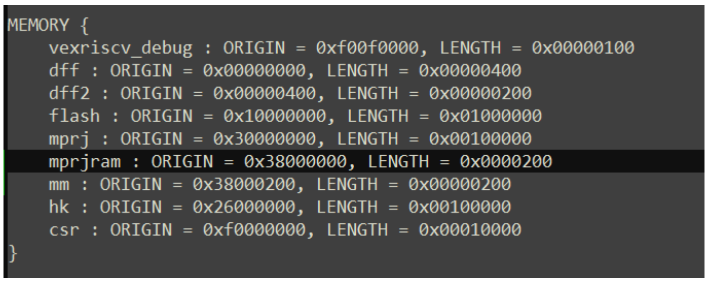
- Introduce the prefetch scheme

做一個prefetch buffer當作cache，Controller不是busy的時候，你就可以先發下一個位置的資料給Controller，那這樣就可以加速它讀回來的時候的時間間隔更短，如果它下一個位置要有資料你已經有在Buffer裡面的話，就可以直接把這個資料回傳回去，就不需要再等待9個T的時間才能讀到下一筆資料。

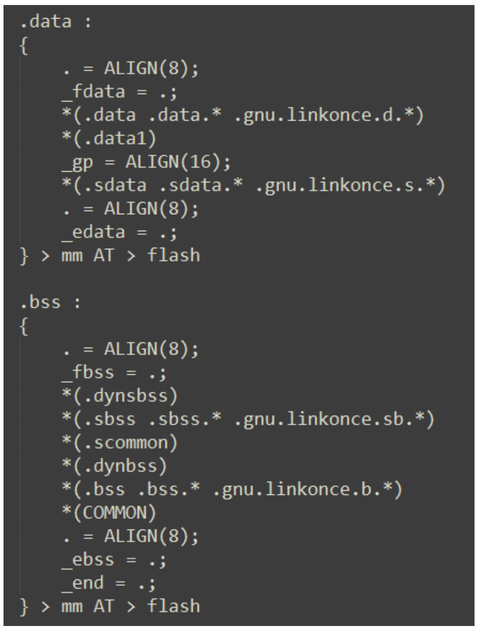
- Introduce the bank interleave for code and data

將code跟data放到不同的bank，比較方便對prefetch的設計，因為可以更好的利用hardware resources減少delay。

- Introduce how to modify the linker to load address/data in two different bank
  我們的做法是把mprjram，拆成兩個length 200的region(mm)，並把data跟bss從原本的dff搬到mm裡。

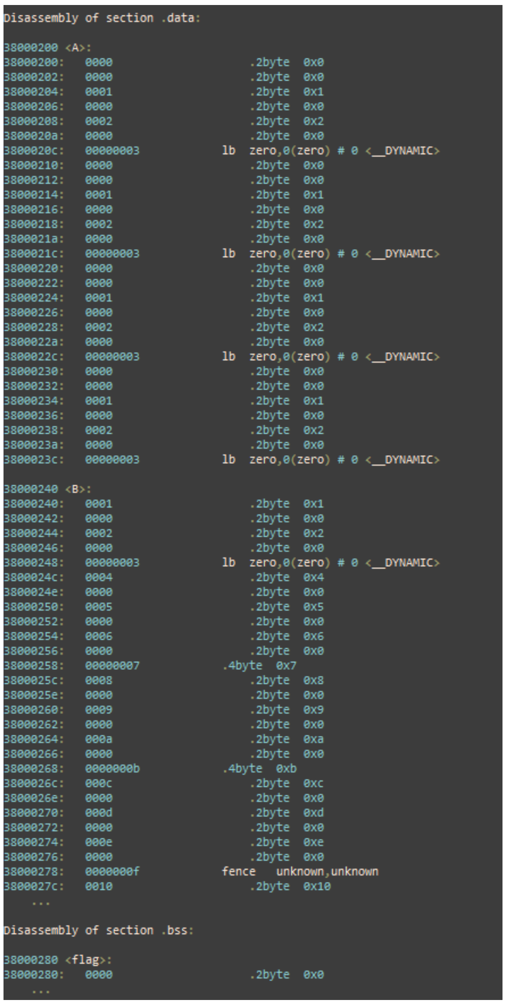


```
MEMORY {
    vexriscv_debug : ORIGIN = 0xf00f0000, LENGTH = 0x00000100
    dff : ORIGIN = 0x00000000, LENGTH = 0x00000400
    dff2 : ORIGIN = 0x00000400, LENGTH = 0x00000200
    flash : ORIGIN = 0x10000000, LENGTH = 0x01000000
    mprj : ORIGIN = 0x30000000, LENGTH = 0x00100000
    mprjram : ORIGIN = 0x38000000, LENGTH = 0x0000200
    mm : 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 = .;
} > mm AT > flash

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

這邊就可以看到矩陣A跟矩陣B的ADDRESS都被放到38000200~38000400

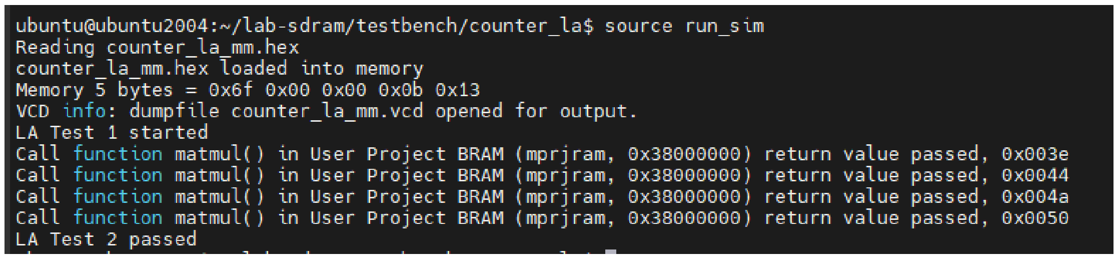


```
Disassembly of section .data:

38000200 <A>:
38000200:   0000                    .2byte  0x0
38000202:   0000                    .2byte  0x0
38000204:   0001                    .2byte  0x1
38000206:   0000                    .2byte  0x0
38000208:   0002                    .2byte  0x2
3800020a:   0000                    .2byte  0x0
3800020c:   00000003            lb  zero,0(zero) # 0 <__DYNAMIC>
38000210:   0000                    .2byte  0x0
38000212:   0000                    .2byte  0x0
38000214:   0001                    .2byte  0x1
38000216:   0000                    .2byte  0x0
38000218:   0002                    .2byte  0x2
3800021a:   0000                    .2byte  0x0
3800021c:   00000003            lb  zero,0(zero) # 0 <__DYNAMIC>
38000220:   0000                    .2byte  0x0
38000222:   0000                    .2byte  0x0
38000224:   0001                    .2byte  0x1
38000226:   0000                    .2byte  0x0
38000228:   0002                    .2byte  0x2
3800022a:   0000                    .2byte  0x0
3800022c:   00000003            lb  zero,0(zero) # 0 <__DYNAMIC>
38000230:   0000                    .2byte  0x0
38000232:   0000                    .2byte  0x0
38000234:   0001                    .2byte  0x1
38000236:   0000                    .2byte  0x0
38000238:   0002                    .2byte  0x2
3800023a:   0000                    .2byte  0x0
3800023c:   00000003            lb  zero,0(zero) # 0 <__DYNAMIC>

38000240 <B>:
38000240:   0001                    .2byte  0x1
38000242:   0000                    .2byte  0x0
38000244:   0002                    .2byte  0x2
38000246:   0000                    .2byte  0x0
38000248:   00000003            lb  zero,0(zero) # 0 <__DYNAMIC>
3800024c:   0004                    .2byte  0x4
3800024e:   0000                    .2byte  0x0
38000250:   0005                    .2byte  0x5
38000252:   0000                    .2byte  0x0
38000254:   0006                    .2byte  0x6
38000256:   0000                    .2byte  0x0
38000258:   00000007            .4byte  0x7
3800025c:   0008                    .2byte  0x8
3800025e:   0000                    .2byte  0x0
38000260:   0009                    .2byte  0x9
38000262:   0000                    .2byte  0x0
38000264:   000a                    .2byte  0xa
38000266:   0000                    .2byte  0x0
38000268:   0000000b            .4byte  0xb
3800026c:   000c                    .2byte  0xc
3800026e:   0000                    .2byte  0x0
38000270:   000d                    .2byte  0xd
38000272:   0000                    .2byte  0x0
38000274:   000e                    .2byte  0xe
38000276:   0000                    .2byte  0x0
38000278:   0000000f            fence   unknown,unknown
3800027c:   0010                    .2byte  0x10
    ...

Disassembly of section .bss:

38000280 <flag>:
38000280:   0000                    .2byte  0x0
    ...
```

- Observe SDRAM access conflicts with SDRAM refresh (reduce the refresh period)

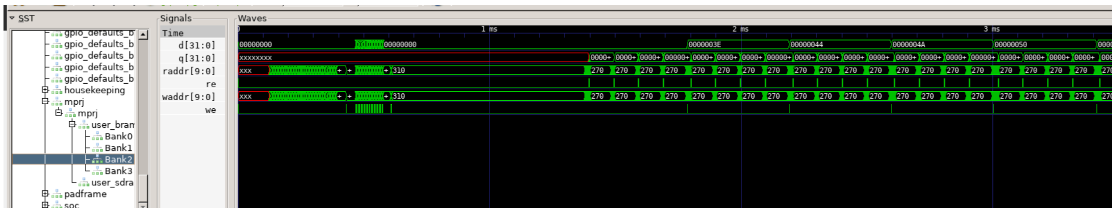
  我們的prefetch功能的第一步是先判斷是code address還是data address，若為code address，則連續讀取八筆資料。但是在新增 prefetch 之後會產生存取延遲。因為如果在進行 refresh 的同時有存取請求，controller 可能需要暫停或延遲某些存取，以確保 refresh 得以進行。

- Simulation Result

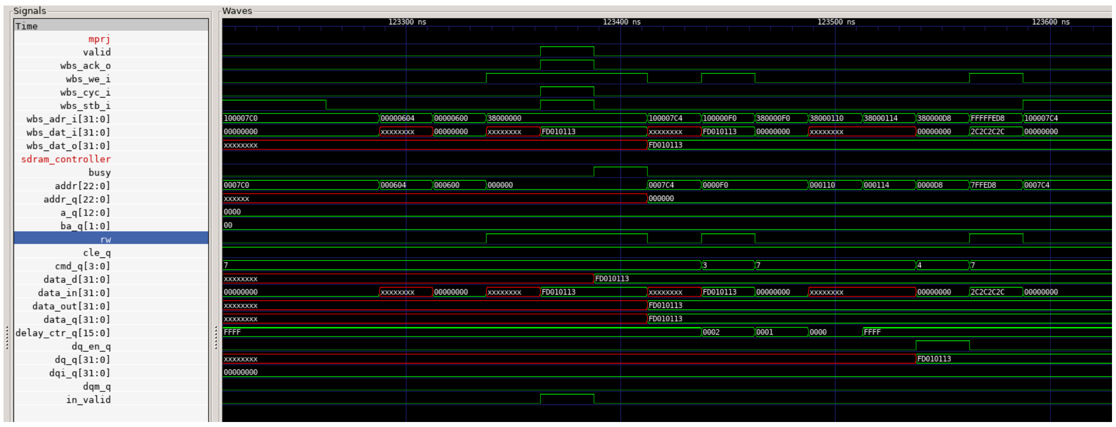
  

```
ubuntu@ubuntu2004:~/lab-sdram/testbench/counter_la$ source run_sim
Reading counter_la_mm.hex
counter_la_mm.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
VCD info: dumpfile counter_la_mm.vcd opened for output.
LA Test 1 started
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x003e
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0044
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x004a
Call function matmul() in User Project BRAM (mprjram, 0x38000000) return value passed, 0x0050
LA Test 2 passed
```

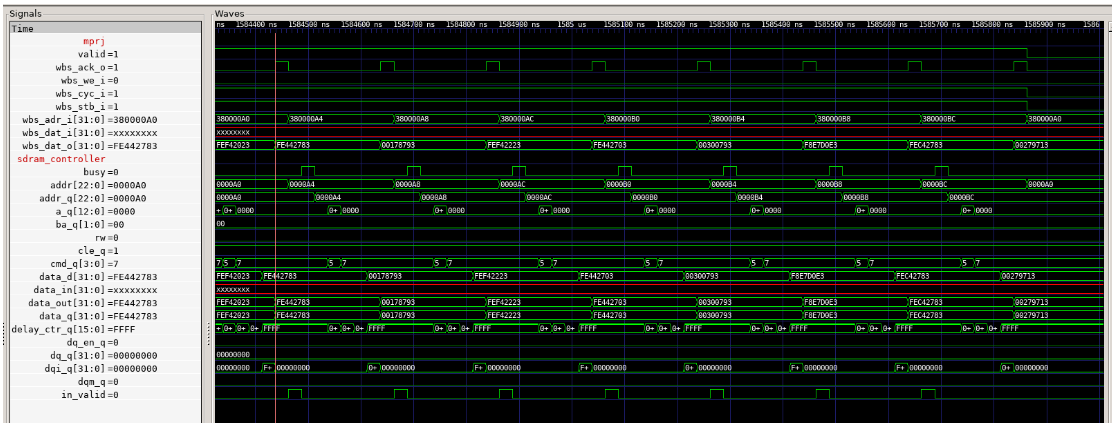
- Waveform
  - bank2 (data)

- write


Signals
Time
mprj
valid
wbs_ack_o
wbs_we_i
wbs_cyc_i
wbs_stb_i
wbs_adr_i[31:0]
wbs_dat_i[31:0]
wbs_dat_o[31:0]
sdram_controller
busy
addr[22:0]
addr_q[22:0]
a_q[12:0]
ba_q[1:0]
rw
cle_q
cmd_q[3:0]
data_d[31:0]
data_in[31:0]
data_out[31:0]
data_q[31:0]
delay_ctr_q[15:0]
dq_en_q
dq_q[31:0]
dqi_q[31:0]
dqm_q
in_valid
Waves
123300 ns
123400 ns
123500 ns
123600 ns

- read


Signals
Time
mprj
valid=1
wbs_ack_o=1
wbs_we_i=0
wbs_cyc_i=1
wbs_stb_i=1
wbs_adr_i[31:0]=380000A0
wbs_dat_i[31:0]=xxxxxxxx
wbs_dat_o[31:0]=FE442783
sdram_controller
busy=0
addr[22:0]=0000A0
addr_q[22:0]=0000A0
a_q[12:0]=0000
ba_q[1:0]=00
rw=0
cle_q=1
cmd_q[3:0]=7
data_d[31:0]=FE442783
data_in[31:0]=xxxxxxxx
data_out[31:0]=FE442783
data_q[31:0]=FE442783
delay_ctr_q[15:0]=FFFF
dq_en_q=0
dq_q[31:0]=00000000
dqi_q[31:0]=00000000
dqm_q=0
in_valid=0
Waves