SOC lab-sdram report

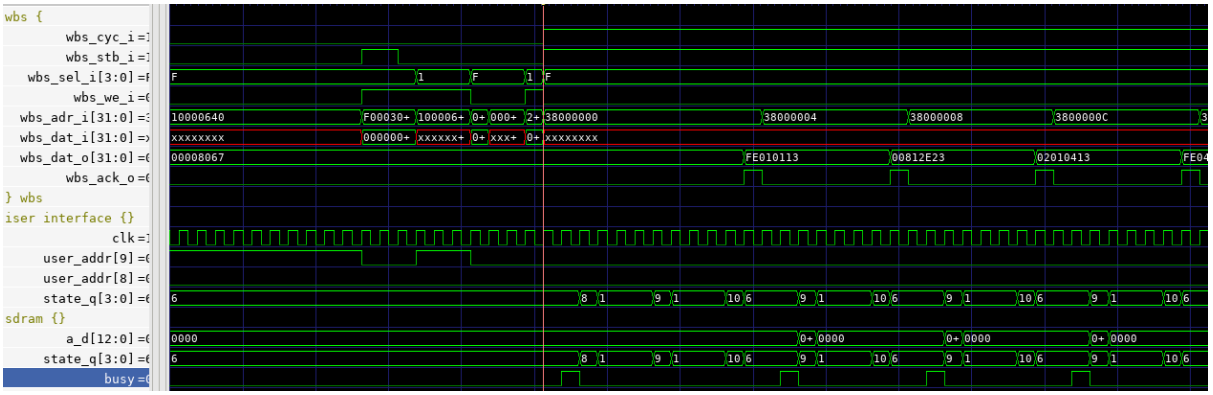
312510145 劉竣瑋
312510147 陳建嘉
312510155 張嘉恩

● the SDRAM controller design, SDRAM bus protocol …

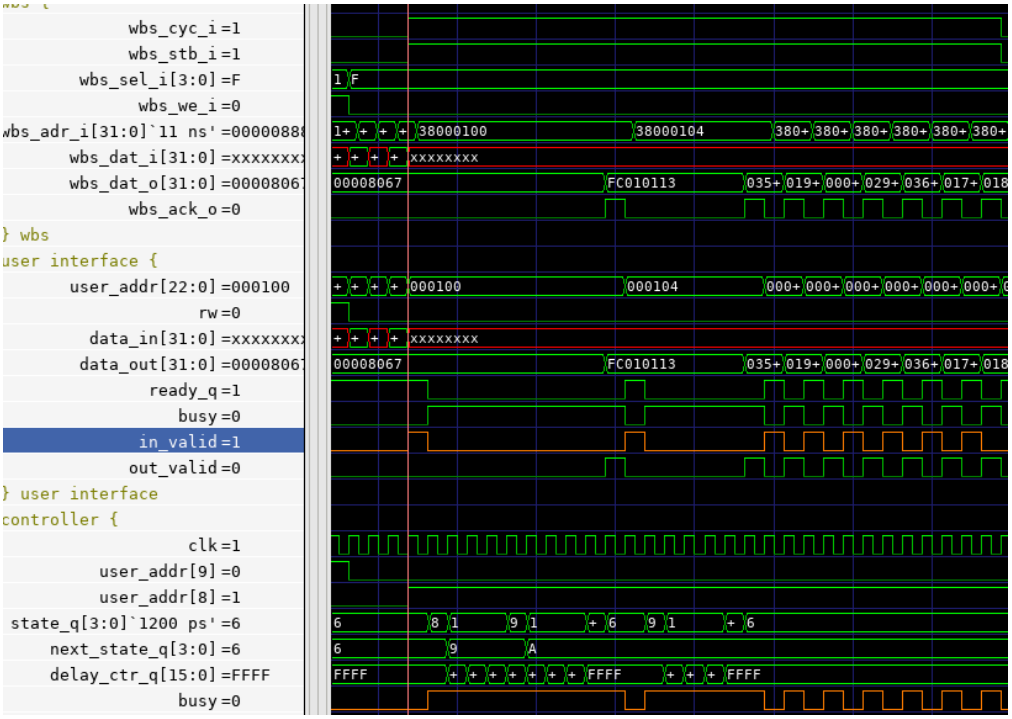
SDRAM controller design:

我們的SDRAM controller design 相較於原本的SDRAM controller design添加了prefetch的功能，下圖是原始波型和prefetch過後的波型

original:



prefetch:



下圖是實現的code

```
// if the row is open we don't have to activate it
if (row_open_q[addr[9:8]]) begin
    if (row_addr_q[addr[9:8]] == addr[22:10]) begin // Row is already open
        if (rw)
            state_d = WRITE;
        //if state is read -> check whether the address is in cache
        else begin
            if (cache_addr_q[addr[2]] == addr[22:0]) begin // if the address is in cache
                out_valid_d = 1'b1;
                data_d = cache_q[addr[2]];
                if(row_open_q[next_addr[9:8]]) begin
                    cmd_d = CMD_READ;
                    a_d = {2'b0, 1'b0, next_addr[7:0], 2'b0};
                    ba_d = next_addr[9:8];
                    cache_addr_d[next_addr[2]] = next_addr;
                    cache_cnt_d[next_addr[2]] = 2;
                end
            end
            else begin
                state_d = READ;
            end
        end
    end
```

```
if (row_open_q[next_addr[9:8]]) begin
    cmd_d = CMD_READ;
    a_d = {2'b0, 1'b0, next_addr[7:0], 2'b0};
    ba_d = next_addr[9:8];
    cache_addr_d[next_addr[2]] = next_addr;
    cache_cnt_d[next_addr[2]] = 2;
end
```

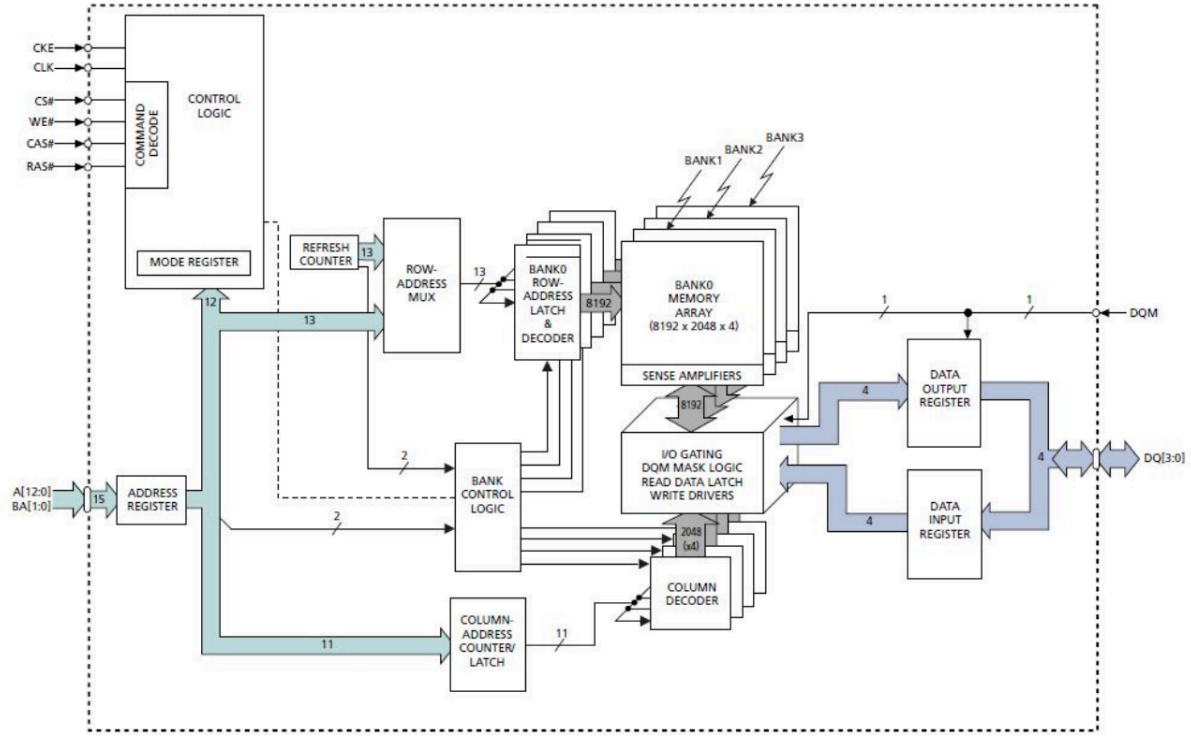
這邊的next_addr會是原本的addr+8，在IDLE收到addr時會去判斷address是否已經在cache裡，如果是的話就會將data直接從cache存取，並送下一筆data的request，進而達到prefetch的目的，減少不必要的latency。

而因為我們針對的是下一筆data的prefetch，因此我們的design並不需要很大的cache size

```
reg [31:0] cache_d[0:1], cache_q[0:1];
reg [22:0] cache_addr_d[0:1], cache_addr_q[0:1];
reg [1:0]  cache_cnt_d[0:1], cache_cnt_q[0:1];
```

如圖所示，我們在實作上只使用了足以存兩筆data的cache，如果今天要對於同一個row去全部做prefetch可能就會需要更大的cache size來實現。

bus protocol:



sdram controller對device的protocol主要由發出去的command控制，得到回傳的sdram_dqi訊號。在device中，sdram_cs對應到command[3],sdram_ras對應到command[2],sdram_cas對應到command[1],sdram_we對應到command[0]。

```
localparam CMD_UNSELECTED    = 4'b1000;
localparam CMD_NOP           = 4'b0111;
localparam CMD_ACTIVE        = 4'b0011;
localparam CMD_READ          = 4'b0101;
localparam CMD_WRITE         = 4'b0100;
localparam CMD_TERMINATE     = 4'b0110;
localparam CMD_PRECHARGE     = 4'b0010;
localparam CMD_REFRESH       = 4'b0001;
localparam CMD_LOAD_MODE_REG = 4'b0000;

// Output assignments
assign sdram_cle = cle_q;
assign sdram_cs = cmd_q[3];
assign sdram_ras = cmd_q[2];
assign sdram_cas = cmd_q[1];
assign sdram_we = cmd_q[0];
assign sdram_dqm = dqm_q;
assign sdram_ba = ba_q;
assign sdram_a = a_q;
```

下面是command的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;
```

● Introduce the prefetch scheme

Prefetching is a technique used in computing to improve the efficiency of data access. It involves preloading data into a cache before it is actually needed. This reduces the time it takes to access the data when it's requested, as it's already available in a faster-access storage.

There are various prefetching schemes:

1. Sequential Prefetching (這邊用的是這個XD) : This is the simplest form, where if data at address 'A' is accessed, the system automatically loads data from subsequent addresses (like A+1, A+2) into the cache, predicting that these will be accessed next.

2. Strided Prefetching: Useful when data accesses have a regular pattern. For example, if a program accesses every nth item in a data set, the prefetcher can load data at intervals of 'n'.

3. Reference-Based Prefetching: This scheme analyzes past memory access patterns and uses this information to predict future accesses.

4. Adaptive Prefetching: This method adjusts its prefetching strategy based on the actual cache hit/miss ratio. It's more complex but can adapt to different usage patterns effectively.

Prefetching can significantly improve performance, especially in systems where data access speed is a bottleneck, such as in large databases or high-performance computing. However, it also has downsides, like potentially increasing cache pollution and memory bandwidth usage.

● Introduce the bank interleave for code and data

下圖是sdr.v裡面的內容

```
blkRam#(.SIZE(mem_sizes), .BIT_WIDTH(DQ_BITS))
Bank0(
    .clk(Sys_clk),
    .we(bwen[0]),
    .re(bren[0]),
    .waddr(Col_brst[9:0]),
    .raddr(Col_brst[9:0]),
    .d(bdi[0]),
    .q(bdq[0])
);
blkRam#(.SIZE(mem_sizes), .BIT_WIDTH(DQ_BITS))
Bank1(
    .clk(Sys_clk),
    .we(bwen[1]),
    .re(bren[1]),
    .waddr(Col_brst[9:0]),
    .raddr(Col_brst[9:0]),
    .d(bdi[1]),
    .q(bdq[1])
);
blkRam#(.SIZE(mem_sizes), .BIT_WIDTH(DQ_BITS))
Bank2(
    .clk(Sys_clk),
    .we(bwen[2]),
    .re(bren[2]),
    .waddr(Col_brst[9:0]),
    .raddr(Col_brst[9:0]),
    .d(bdi[2]),
    .q(bdq[2])
);
blkRam#(.SIZE(mem_sizes), .BIT_WIDTH(DQ_BITS))
Bank3(
    .clk(Sys_clk),
    .we(bwen[3]),
    .re(bren[3]),
    .waddr(Col_brst[9:0]),
    .raddr(Col_brst[9:0]),
    .d(bdi[3]),
    .q(bdq[3])
);
```

可以看到不同的bank之間是獨立運作的，不同bank之間是能夠獨立被存取的，因此在本次lab中我們將load跟data存放在不同的bank中進而實現bank interleave.

● Introduce how to modify the linker to load address/data in two different bank

這邊是我們的memory mapping

```
MEMORY {
    vexriscv_debug : ORIGIN = 0xf00f0000, LENGTH = 0x00000100
    dff : ORIGIN = 0x00000000, LENGTH = 0x000000100
    dff2 : ORIGIN = 0x00000400, LENGTH = 0x00000200
    flash : ORIGIN = 0x10000000, LENGTH = 0x01000000
    mprj : ORIGIN = 0x30000000, LENGTH = 0x00100000
    mprjram : ORIGIN = 0x38000100, LENGTH = 0x00000200
    hk : ORIGIN = 0x26000000, LENGTH = 0x00100000
    csr : ORIGIN = 0xf0000000, LENGTH = 0x00010000
}
```

data(matrix A,B)的部分可以看到會落在dff的部分(0x00000000-0x00000080)，而workload則是被放在0x38000100的位置，因此我們可以透過第user_addr[8]來將data跟inst存放在不同的bank中，如下圖所示:

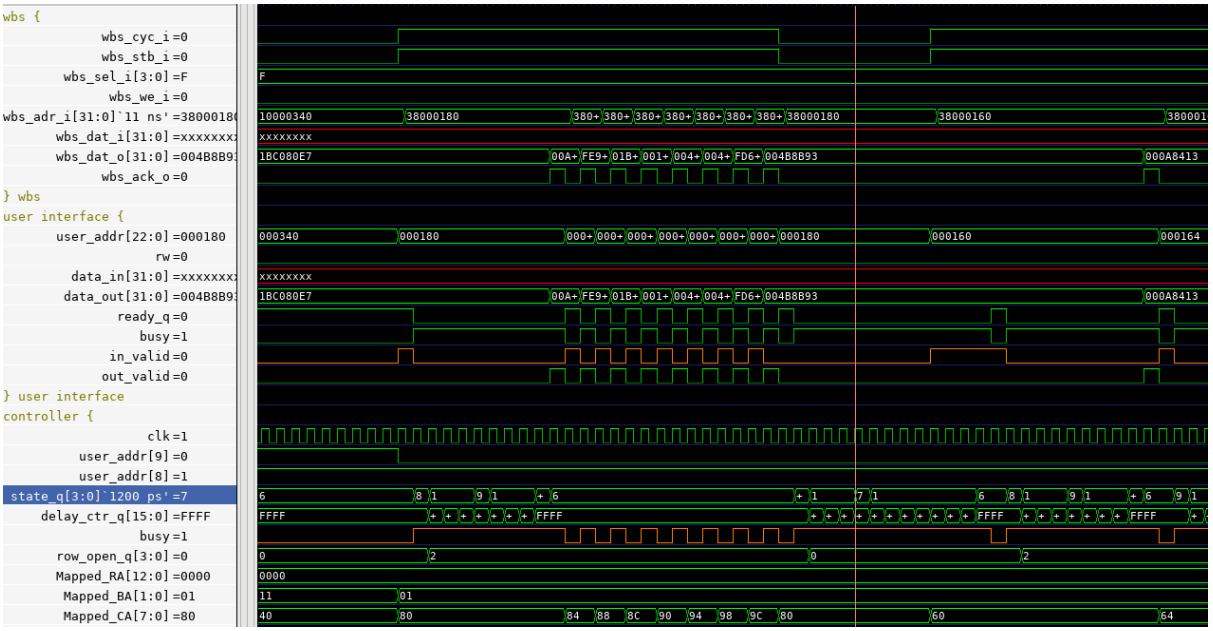
```
assign Mapped_RA = {user_addr[22:10]};
assign Mapped_BA = {user_addr[9:8]};
assign Mapped_CA = {user_addr[7:0]};
```

這樣data就會被存放在BA = 2'b00的位置。

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

```
// SDRAM Timing
localparam tCASL            = 13'd2;       // 3T actually
localparam tPRE             = 13'd2;       // 3T
localparam tACT             = 13'd2;       // 3T
localparam tREF             = 13'd6;       // 7T
localparam tRef_Counter     = 10'd50;     //
```

我們將refresh period設為50T進行觀察



從波型上可以發現，當我們在讀instruction open row的時候遇到refresh時，controller的state會進行3 cycle latency的precharge和7 cycle latency的refresh，之後會再activate一次row後才能繼續之後的讀取。