

# DRAM module for RTL code

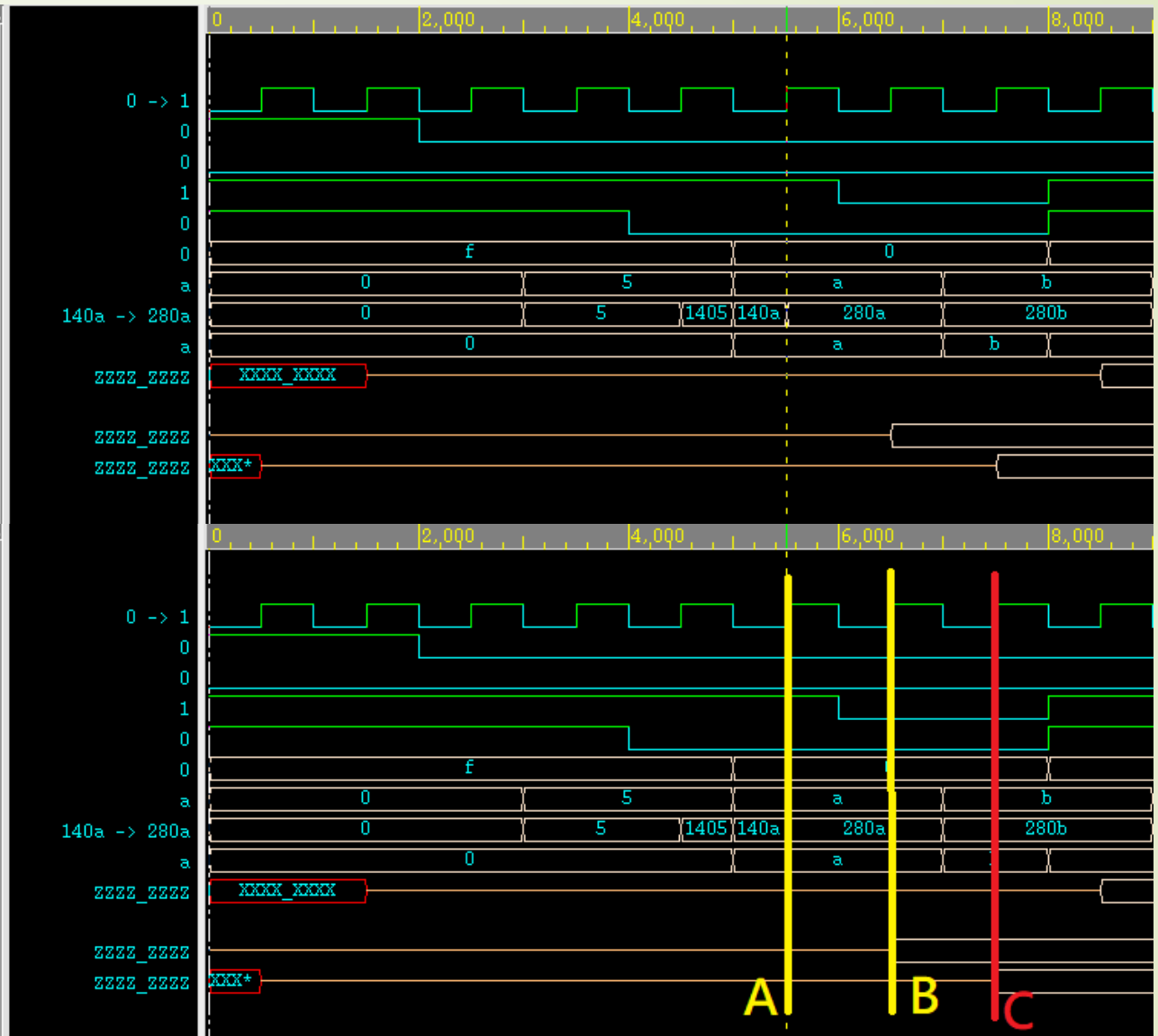
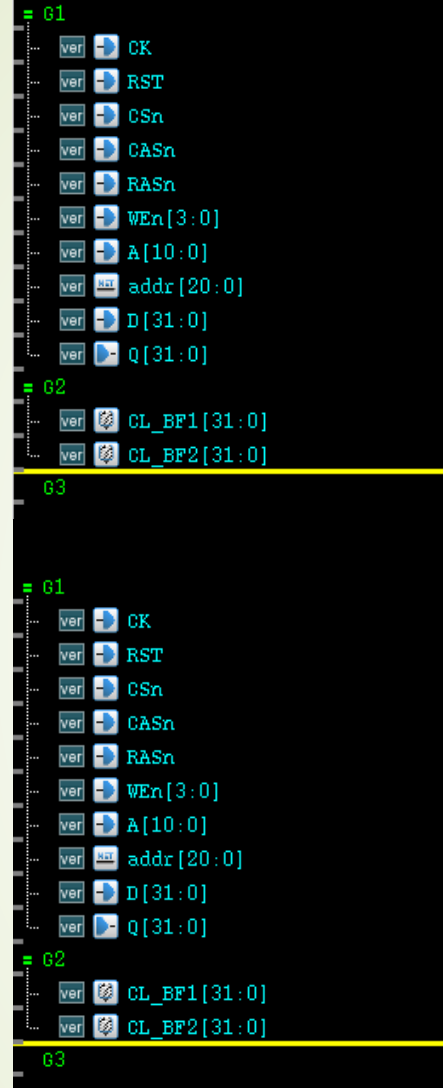
- ◆ RTL code 可以使用Verilog或SystemVerilog撰寫。
- ◆ RTL code所採用的環境可以是任何環境，包含NC-Verilog、Modelsim、Xilinx等Toolchain，DEMO時必須攜帶筆電使用自己環境呈現。
- ◆ 提供的DRAM\_tb.sv和Makefile是使用NC-Verilog及Verdi環境，如需使用其他環境則必須修改DRAM\_tb.sv程式中fsdbDump的部分。
- ◆ 下頁開始是介紹DRAM 的行為。

# DRAM (1)

DRAM	System signals			
	CK	input	1	System clock
	RST	input	1	System reset(active high)
	Memory ports			
	<u>CSn</u>	input	1	DRAM Chip Select (active low)
	Wen	input	4	DRAM Write Enable (active low)
	<u>RASn</u>	input	1	DRAM ROW Access strobe (active low)
	<u>CASn</u>	input	1	DRAM Column Access strobe (active low)
	A	input	11	DRAM Address input
	D	input	32	DRAM data input
	Q	output	32	DRAM data output
	Memory space			
	Memory_byte0	reg	8	Size : [0:2097151]
	Memory_byte1	reg	8	Size : [0:2097151]
	Memory_byte2	reg	8	Size : [0:2097151]
	Memory_byte3	reg	8	Size : [0:2097151]

※ Row address is 11-bit and column address is 10-bit

# DRAM (2)

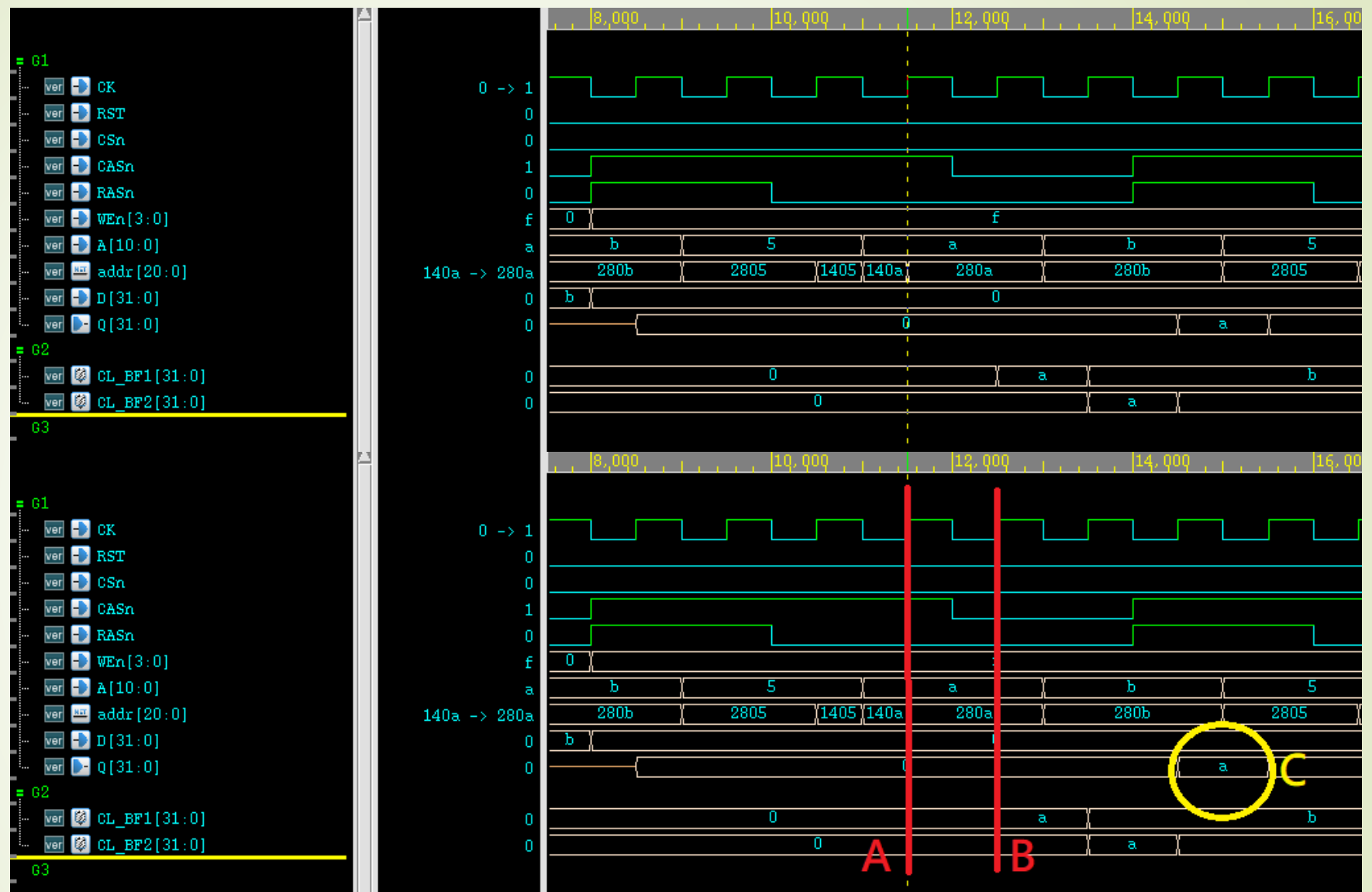


✖ Write

A. RAS=0 -> Set Row (A should be ready)

B. CAS=0 -> Set Column(A, WEn and D should be ready)

# DRAM (3)



✖ Read

A. RAS=0 -> Set Row (A should be ready)

B. CAS=0 -> Set Column(A, WEn should be ready)

C. Latency : 2 cycle

# DRAM module for RTL code

- ◆ DRAM\_INPUT資料夾內有4個檔案，會一起放入DRAM。

input_0.hex	2020/5/5 下午 03:15	HEX 檔案	1,587 KB
input_1.hex	2020/5/5 下午 03:15	HEX 檔案	1,587 KB
input_2.hex	2020/5/5 下午 03:15	HEX 檔案	1,587 KB
input_3.hex	2020/5/5 下午 03:15	HEX 檔案	1,587 KB

- ◆ 在top\_tb.sv中會利用readmemh將資料讀入，下圖紅框中，因為此範例是在top\_tb.sv宣告DRAM module，所以後方讀入的位置是這樣表示。如果你是在top.sv宣告DRAM module，則根據你的instance name會有不同寫法，綠色的註解為範例。

```
$readmemh({prog_path, "/input_0.hex"}, M1.Memory_byte0);  
$readmemh({prog_path, "/input_1.hex"}, M1.Memory_byte1);  
$readmemh({prog_path, "/input_2.hex"}, M1.Memory_byte2);  
$readmemh({prog_path, "/input_3.hex"}, M1.Memory_byte3);  
  
// $readmemh({prog_path, "/input_feature0.hex"}, top_1.DRAM_1.Memory_byte0);  
// $readmemh({prog_path, "/input_feature1.hex"}, top_1.DRAM_1.Memory_byte1);  
// $readmemh({prog_path, "/input_feature2.hex"}, top_1.DRAM_1.Memory_byte2);  
// $readmemh({prog_path, "/input_feature3.hex"}, top_1.DRAM_1.Memory_byte3);
```

# DRAM module for RTL code

- ◆ Input feature的起始點為'h000000，Weight的起始點為'h100000。
- ◆ 下圖紅框讓我們可以直接在top\_tb內存取DRAM的值，可以更容易Debug。

```
`define INPUT_START 'h0000
`define WEIGHT_START 'h100000

`define mem_word(addr){M1.Memory_byte3[addr],M1.Memory_byte2[addr],M1.Memory_byte1[addr],M1.Memory_byte0[addr]}
```

- ◆ 下圖會印出前60個的Input feature值，而Input feature總共有 $416*416*3$ ，Weight則會有 $3*3*3*16$ ，每個值皆為32bit。

```
initial begin
    #(`CYCLE*50000)
    for(i=0;i<60;i++)    //print first 60 value
    begin
        $display("%h",`mem_word(`INPUT_START + i));
    end
    $finish;
end
```

# DRAM module for RTL code

- ◆ DRAM裡使用4個reg array，因為實際上DRAM是可以針對特定的Byte去做存取，Wen訊號也因此為4bit。
- ◆ 但在作業中我們所使用的資料為32bit，存放在4個reg array中的同一個位置裡，所以只要同時操作4個Byte存或寫，並不需要將32bit分成4個Byte操作，具體來說就是存值時Wen=0，讀值時Wen=F。

DRAM	System signals			
	CK	input	1	System clock
	RST	input	1	System reset(active high)
	Memory ports			
	<u>C</u> Sn	input	1	DRAM Chip Select (active low)
	Wen	input	4	DRAM Write Enable (active low)
	<u>R</u> ASn	input	1	DRAM ROW Access strobe (active low)
	<u>C</u> ASn	input	1	DRAM Column Access strobe (active low)
	A	input	11	DRAM Address input
	D	input	32	DRAM data input
	Q	output	32	DRAM data output
	Memory space			
	Memory_byte0	reg	8	Size : [0:2097151]
	Memory_byte1	reg	8	Size : [0:2097151]
	Memory_byte2	reg	8	Size : [0:2097151]
	Memory_byte3	reg	8	Size : [0:2097151]

※ Row address is 11-bit and column address is 10-bit