FPGA_HW4 設計說明

第14組

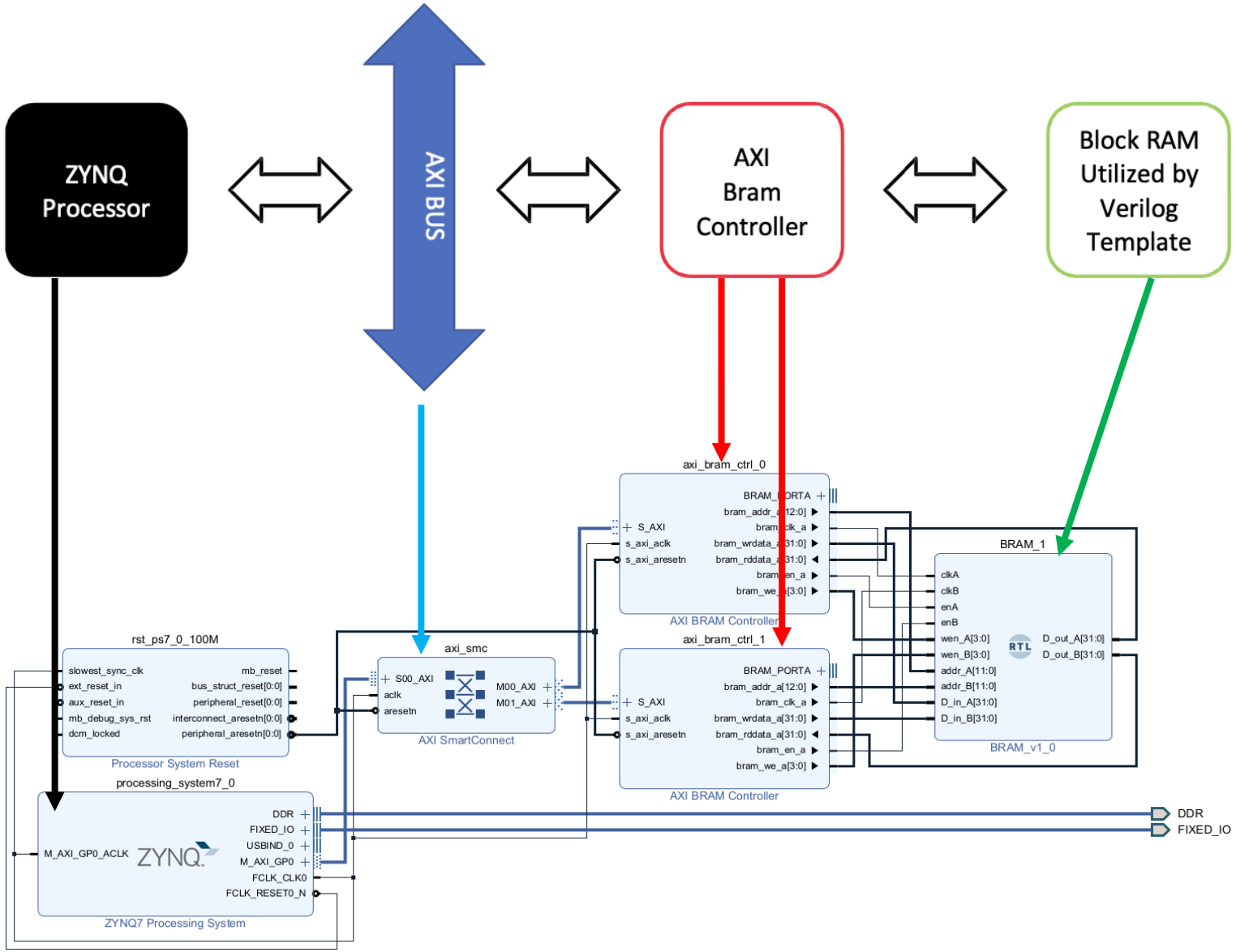
黃偉峰 E34106010

陳識博 E94106096

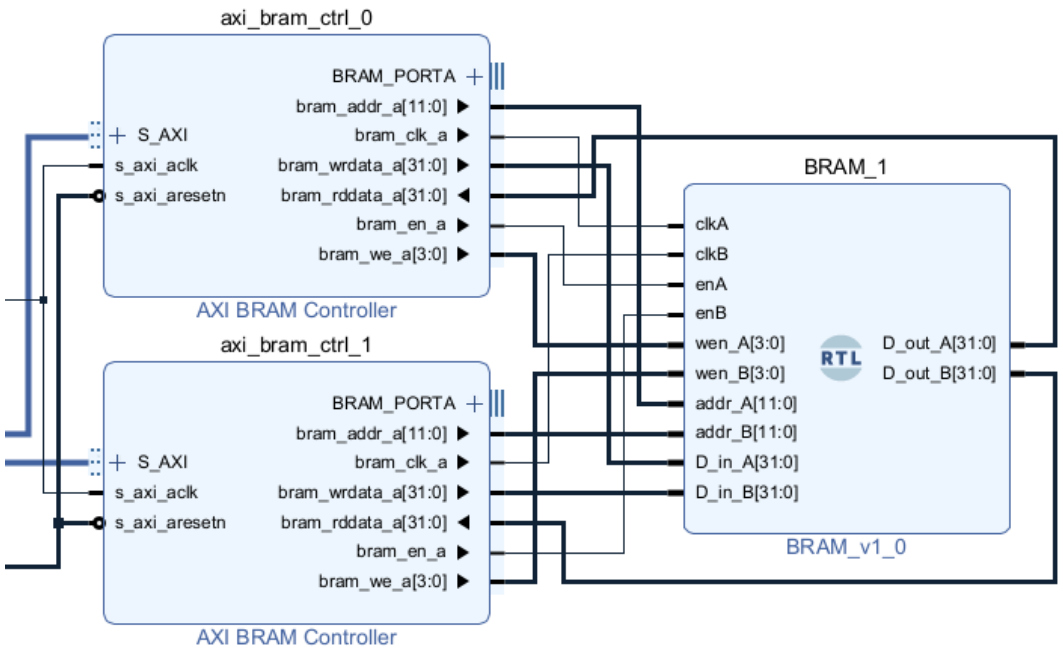
黃芊 F74104040

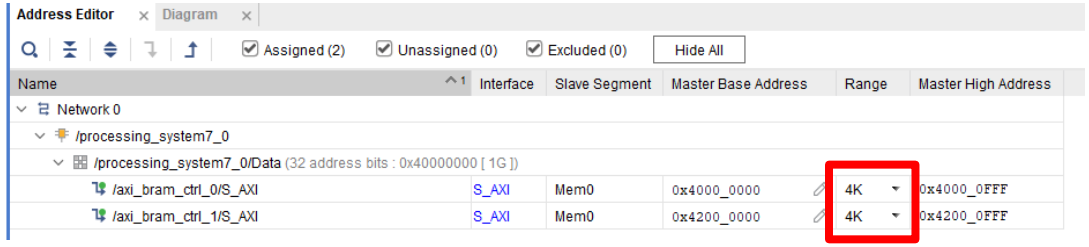
Problem 1 – Block RAM Utilize

Block Diagram



Bram controller 配置



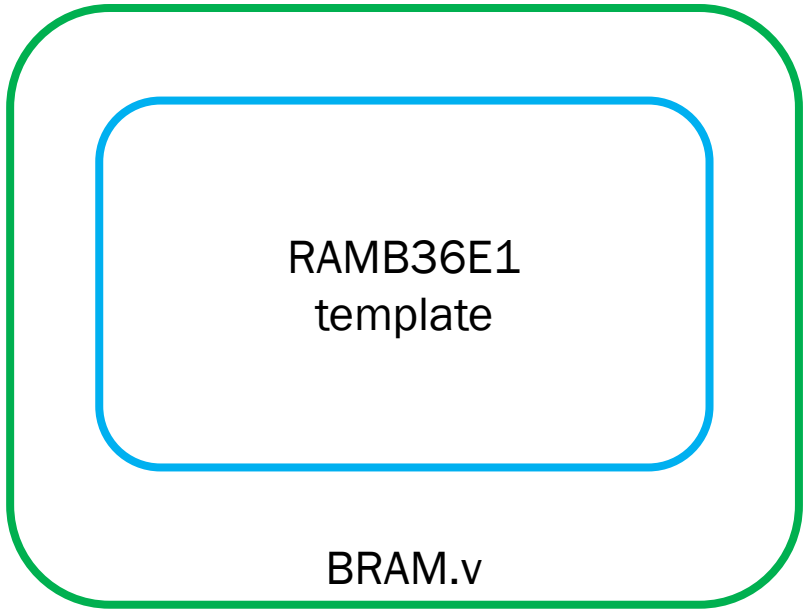


AXI BRAM controller只會被分配到一個Address，只能控制一個port，因此本次設計為True Dual Port 的情況下需使用到兩顆AXI BRAM Controller，這樣子才能獲得兩個Address來給BRAM的Port A、Port B做使用。

AXI protocol為Byte Addressable的，而我們要控制的BRAM為32Kb的大小，換算成Byte為32Kb / 8bit = 4KB，因此在Range設定這邊需使用4K。

根據address path diagram可以看到，分配給AXI Bram Controller的0x4000_0000~0x4000_0FFF經過AXI Smart Connect後就會變成0x0~0xFFF，給後續32Kb(4KB)的BRAM做使用。

Bram_v1_0 IP 設計



SPEC	
Data width	32bit
Memory Size	32Kb
RAM Mode	TDP
Initial Contents	Offset = 0：0x2597 Offset = 4：0x6425 Offset = 28：0x5071 Offset = 64：0x8CF5

Bram_v1_0 IP內部使用RAMB36E1 template來初始化並根據助教提供的SPEC來做參數調整，後續將介紹內部檔案設置的各項參數。

```
module BRAM(
    input clkA,
    input clkB,
    input enA,
    input enB,
    input [3:0] wen_A,
    input [3:0] wen_B,
    input [11:0] addr_A,
    input [11:0] addr_B,
    input [31:0] D_in_A,
    input [31:0] D_in_B,
    output [31:0] D_out_A,
    output [31:0] D_out_B
);
```

Addr Width為12bit，因為AXI protocol為Byte Addressable的，32Kb的記憶體需要用到4K的range去做定址，因此為12bit width。

Data Width為32bit，因此input(D_in_X)、output(D_out_X)的部分都設置為32bit

RAMB36E1 Available Attributes

```
// RAM Mode: "SDP" or "TDP"
.RAM_MODE("TDP"),
```

根據SPEC要求，將RAM_MODE設置為True Dual Port mode

```
// Address Collision Mode: "PERFORMANCE" or "DELAYED_W
.RDADDR_COLLISION_HWCONFIG("DELAYED_WRITE"),
// Collision check: Values ("ALL", "WARNING_ONLY", "GE
.SIM_COLLISION_CHECK("ALL"),
// DOA_REG, DOB_REG: Optional output register (0 or 1)
.DOA_REG(1),
.DOB_REG(1),

.EN_ECC_READ("FALSE"),
.EN_ECC_WRITE("FALSE"),
```

DOA_REG、DOB_REG都設成1，在RAMB36E1的output將會有一個額外的Register拿來儲存輸出的值，讓後續的電路可以有一個完整的cycle來做存取資料。

ECC設置皆設定成FALSE，因RAMB36E1設置成TDP模式時不能使用ECC。

RAMB36E1 Available Attributes

```
// READ_WIDTH_A/B, WRITE_WIDTH_A/B: Read/write width per port
.READ_WIDTH_A(36),                                          // 0-72
.READ_WIDTH_B(36),                                          // 0-36
.WRITE_WIDTH_A(36),                                         // 0-36
.WRITE_WIDTH_B(36),                                         // 0-72
```

Table 1-13: **Port Aspect Ratio for RAMB36E1 (in TDP Mode)**

Port Data Width	Port Address Width	Depth	ADDR Bus	DI Bus DO Bus	DIP Bus DOP Bus
1	15	32,768	[14:0]	[0]	NA
2	14	16,384	[14:1]	[1:0]	NA
4	13	8,192	[14:2]	[3:0]	NA
9	12	4,096	[14:3]	[7:0]	[0]
18	11	2,048	[14:4]	[15:0]	[1:0]
36	10	1,024	[14:5]	[31:0]	[3:0]
1 (Cascade)	16	65536	[15:0]	[0]	NA

RAMB36E1在TDP模式下，Port Data Width只能設置成上方幾種數字，此次設計為read/write width為32bit的BRAM，因此設置成36bit，36bit其中32bit為data另外4個bit為Parity bit，雖然TDP下不支援ECC但根據xilinx提供的文檔我們不能單純設置成32bit仍需設置成36bit才正確。

RAMB36E1 Available Attributes

第七個Word　　第二個Word　第一個Word

```
// INIT_00 to INIT_7F: Initial contents of the data memory array
.INIT_00(256'h00005071_00000000_00000000_00000000_00000000_00000000_00006425_00002597),
.INIT_01(256'h00000000_00000000_00000000_00000000_00000000_00000000_00000000_00000000),
.INIT_02(256'h00000000_00000000_00000000_00000000_00000000_00000000_00000000_00008CF5),
```

SPEC	
Initial Contents	Offset = 0：0x2597 Offset = 4：0x6425 Offset = 28：0x5071 Offset = 64：0x8CF5

依題目要求，每32 bit為一個word。Offset則為從Base address往後幾個Byte，其中每4個offset代表4 byte，亦即1 word。E.g. offset=28，則將數值設置在第7個word上。

RAMB36E1 Port Descriptions

```
.CLKARDCLK(clkA),          // 1-bit input: A port clock/Read clock
.CLKBWRCLK(clkB),          // 1-bit input: B port clock/Write clock
```

RAMB36E1 在 True Dual Port (TDP) 模式下支援 asynchronous clock domains，Port A、Port B可以使用不同的clock。

```
.ENARDEN(enA),              // 1-bit input: A port enable/Read enable
.ENBWREN(enB),              // 1-bit input: B port enable/Write
.WEA(wen_A),                    // 4-bit input: A port write enable
.WEBWE({4'b0000, wen_B}),                   // 8-bit input: B port write enable/Write enable
```

Enable	WriteEnable	實際情況
0	0	不能讀寫
0	1	不能讀寫
1	0	可讀
1	1	可寫

因RAMB36E1支援Byte wise write，因此WriteEnable訊號為4bit，因為32bit width為4Byte，WriteEnable的4個bit各控制其中一個Byte的讀寫。

E.g.:若只要更新第一個byte，則WEA將被設定成4’b0001。

RAMB36E1 Available Attributes

```
.REGCEAREGCE(1'b1),     // 1-bit input: A port register enable/Register enable
.REGCEB(1'b1),              // 1-bit input: B port register enable
```

先前在avaliable attribute時，我們有設定output register，而上圖兩個訊號則控制該output register是否去做更新，因此都設為1，使他保持更新output register。

```
.DOADO(D_out_A),                // 32-bit output: A port data/LSB data
.DOBDO(D_out_B),                // 32-bit output: B port data/MSB data
.DIADI(D_in_A),                // 32-bit input: A port data/LSB data
.DIBDI(D_in_B)                // 32-bit input: B port data/MSB data
```

RAM input/output data皆為32-bit，而DOADO()、DOBDO()為output dual port，DIADI()、DIBDI()為input dual port。

RAMB36E1 Available Attributes

```
.ADDRBWRADDR({1'b0, addr_B[11:2], 5'b00000}),     // 16-bit input: B port address/Write address
.ADDRARDADDR({1'b0, addr_A[11:2], 5'b00000}),     // 16-bit input: A port address/Read address
```

RAMB36E1 實際上包含 36Kb 的容量，其中 32Kb 用於資料儲存（Data Bits），額外的 4Kb 則專門用來儲存Parity Bits，這些位元只會在啟用 ECC 模式時使用。

RAMB36E1 支援最小資料寬度為 1-bit，因此若配置為 1-bit 模式，便需要 2^15個地址才能存滿 32Kb。Address Width之所以設定為 16-bit，是因為：

- 實際地址需要 15-bit
- 最前面保留的 1-bit 是用來作為 cascade 模式（串接多個 BRAM）時的選擇bit

因此總共需要 16-bit 的地址輸入。

在使用 AXI BRAM Controller 時，addr_A 提供的是 Byte Address， 而 RAMB36E1 實際以每 1 word = 32 bits = 4 bytes 為單位存取。因此在連接時需將最低 0 ~ 1 位元忽略，地址從 addr_A[11:2] 開始使用， 此處的位元表示的是第幾個 word 單位的地址。

RAMB36E1 Available Attributes

Table 1-13: **Port Aspect Ratio for RAMB36E1 (in TDP Mode)**

Port Data Width	Port Address Width	Depth	ADDR Bus	DI Bus DO Bus	DIP Bus DOP Bus
1	15	32,768	[14:0]	[0]	NA
2	14	16,384	[14:1]	[1:0]	NA
4	13	8,192	[14:2]	[3:0]	NA
9	12	4,096	[14:3]	[7:0]	[0]
18	11	2,048	[14:4]	[15:0]	[1:0]
36	10	1,024	[14:5]	[31:0]	[3:0]
1 (Cascade)	16	65536	[15:0]	[0]	NA

```
.ADDRBWRADDR({1'b0, addr_B[11:2], 5'b00000}),     // 16-bit input: B port address/Write address
.ADDRARDADDR({1'b0, addr_A[11:2], 5'b00000}),     // 16-bit input: A port address/Read address
```

根據xilinx提供的BRAM配置文檔可以看到，Port Address Width = 10bit對應到我們設置的addr_A[11:2]，ADDR Bus = [14:5]則對應到.ADDRARDADDR裡面的第14個bit到第2個bit剛好為addr_A[11:2]的位子。

Utilization of BRAM

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	581	53200	1.09
LUTRAM	2	17400	0.01
FF	736	106400	0.69
BRAM	1	140	0.71
BUFG	1	32	3.13

根據Utilization report可以看到Pynq-z2的BRAM共有140個可使用，而我們這次透過RAMB36E1 template 實例化了一顆dual port BRAM使用了其中的1個BRAM。因此從上方utilization report看到我們BRAM用了1/140 = 0.71%。

Software Testing

讀取預先Initialize在BRAM中的Data:

```
xil_printf("Preset data read:\r\n");
for (int i = 0; i < 4; ++i) {
    u32 d = Xil_In32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + preset_offset[i]);
    xil_printf("Read  0x%08lx @ +%d\r\n", d, preset_offset[i]);
}
```

透過Port A寫入後再透過Port A讀取:

```
xil_printf("\r\nPort A read after write:\r\n");
for (int i = 0; i < 5; ++i) {
    Xil_Out32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + offsets[i], pattern[i]);
    xil_printf("Port A Write 0x%08lx @ +%d\r\n", pattern[i], offsets[i]);
}

xil_printf("\r\nPort A Read back\r\n");
for (int i = 0; i < 5; ++i) {
    u32 d = Xil_In32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + offsets[i]);
    xil_printf("Port A Read 0x%08lx @ +%d\r\n", d, offsets[i]);
}
```

Software Testing

透過Port B寫入後再透過Port B讀取:

```
xil_printf("\r\nPort B read after write:\r\n");
for (int i = 0; i < 5; ++i) {
    Xil_Out32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + offsets[i], pattern[4 - i]);
    xil_printf("Port B Write 0x%08lx @ +%d\r\n", pattern[4 - i], offsets[i]);
}

xil_printf("\r\nPort B Read back\r\n");
for (int i = 0; i < 5; ++i) {
    u32 d = Xil_In32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + offsets[i]);
    xil_printf("Port B Read 0x%08lx @ +%d\r\n", d, offsets[i]);
}
```

Software Testing

Port A做MARCH C algorithm:

```
  xil_printf("\r\n--- Start March C (Port A) ---\r\n");
// ↑ (w0)
for (int i = 0; i < 1024; ++i) {
    Xil_Out32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + i * 4, 0x00000000);
}

// ↑ (r0, w1)
for (int i = 0; i < 1024; ++i) {
    u32 d = Xil_In32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + i * 4);
    if (d != 0x00000000) {
        xil_printf("Port A MarchC ERROR ↑r0 @ +%d: read 0x%08lx\r\n", i * 4, d);
    }
    Xil_Out32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + i * 4, 0xFFFFFFFF);
}

// ↓ (r1, w0)
for (int i = 1023; i >= 0; --i) {
    u32 d = Xil_In32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + i * 4);
    if (d != 0xFFFFFFFF) {
        xil_printf("Port A MarchC ERROR ↓r1 @ +%d: read 0x%08lx\r\n", i * 4, d);
    }
    Xil_Out32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + i * 4, 0x00000000);
}

// ↑ (r0)
for (int i = 0; i < 1024; ++i) {
    u32 d = Xil_In32(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR + i * 4);
    if (d != 0x00000000) {
        xil_printf("Port A MarchC ERROR ↑r0 final @ +%d: read 0x%08lx\r\n", i * 4, d);
    }
}
xil_printf("--- Port A March C Done ---\r\n");
```

Software Testing

Port B做MARCH C algorithm:

```
xil_printf("\r\n--- Start March C (Port B) ---\r\n");
// ↑ (w0)
for (int i = 0; i < 1024; ++i) {
    Xil_Out32(XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + i * 4, 0x00000000);
}

// ↑ (r0, w1)
for (int i = 0; i < 1024; ++i) {
    u32 d = Xil_In32(XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + i * 4);
    if (d != 0x00000000) {
        xil_printf("MarchC ERROR ↑r0 fail @ +%d: read 0x%08lx\r\n", i * 4, d);
    }
    Xil_Out32(XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + i * 4, 0xFFFFFFFF);
}

// ↓ (r1, w0)
for (int i = 1023; i >= 0; --i) {
    u32 d = Xil_In32(XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + i * 4);
    if (d != 0xFFFFFFFF) {
        xil_printf("MarchC ERROR ↓r1 fail @ +%d: read 0x%08lx\r\n", i * 4, d);
    }
    Xil_Out32(XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + i * 4, 0x00000000);
}

// ↑ (r0)
for (int i = 0; i < 1024; ++i) {
    u32 d = Xil_In32(XPAR_AXI_BRAM_CTRL_1_S_AXI_BASEADDR + i * 4);
    if (d != 0x00000000) {
        xil_printf("MarchC ERROR ↑r0 final fail @ +%d: read 0x%08lx\r\n", i * 4, d);
    }
}
xil_printf("--- Port B March C Done ---\r\n");
```

Software Testing Result

```
Preset data read:
Read  0x00002597 @ +0
Read  0x00006425 @ +4
Read  0x00005071 @ +28
Read  0x00008CF5 @ +64

Port A read after write:
Port A Write 0x00002597 @ +8
Port A Write 0x00006425 @ +12
Port A Write 0x00005071 @ +36
Port A Write 0x00008CF5 @ +72
Port A Write 0x0000FFFF @ +4092

Port A Read back
Port A Read 0x00002597 @ +8
Port A Read 0x00006425 @ +12
Port A Read 0x00005071 @ +36
Port A Read 0x00008CF5 @ +72
Port A Read 0x0000FFFF @ +4092

Port B read after write:
Port B Write 0x0000FFFF @ +8
Port B Write 0x00008CF5 @ +12
Port B Write 0x00005071 @ +36
Port B Write 0x00006425 @ +72
Port B Write 0x00002597 @ +4092

Port B Read back
Port B Read 0x0000FFFF @ +8
Port B Read 0x00008CF5 @ +12
Port B Read 0x00005071 @ +36
Port B Read 0x00006425 @ +72
Port B Read 0x00002597 @ +4092

--- Start March C (Port A) ---
--- Port A March C Done ---

--- Start March C (Port B) ---
--- Port B March C Done ---
```

Problem2 – Q&A

1.PYNQ-Z2 上共有多少容量的Block RAM?

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	581	53200	1.09
LUTRAM	2	17400	0.01
FF	736	106400	0.69
BRAM	1	140	0.71
BUFG	1	32	3.13

Ans:PYNQ-Z2上總共有140顆BRAM，而每顆BRAM為36Kb，因此總共的容量為36*1024*140=5160960bits = 4.921875Mb。

1.PYNQ-Z2 上共有多少容量的Block RAM?

	Cost-Optimized Devices					
Device Name	Z-7007S	Z-7012S	Z-7014S	Z-7010	Z-7015	Z-7020
Part Number	XC7Z007S	XC7Z012S	XC7Z014S	XC7Z010	XC7Z015	XC7Z020
Processor Core	Single-Core ARM® Cortex™-A9 MPCore™ Up to 766MHz			Dual-Core ARM Cortex-A9 MPCore Up to 866MHz		
Processor Extensions	NEON™ SIMD Engine and Single/Double Precis					
L1 Cache	32KB Instruction, 32KB Da					
L2 Cache	512KB					
On-Chip Memory	256KB					
External Memory Support[2]	DDR3, DDR3L, DDR					
External Static Memory Support[2]	2x Quad-SPI, NAN					
DMA Channels	8 (4 dedicated					
Peripherals	2x UART, 2x CAN 2.0B, 2x I2C,					
Peripherals w/ built-in DMA[2]	2x USB 2.0 (OTG), 2x Tri-mode Giga					
Security[3]	RSA Authentication of First S AES and SHA 256b Decryption and Au					
Processing System to Programmable Logic Interface Ports (Primary Interfaces & Interrupts Only)	2x AXI 32b Master, 2x 4x AXI 64b/32b M AXI 64b AC 16 Interrup					
7 Series PL Equivalent	Artix®-7	Artix-7	Artix-7	Artix-7	Artix-7	Artix-7
Logic Cells	23K	55K	65K	28K	74K	85K
Look-Up Tables (LUTs)	14,400	34,400	40,600	17,600	46,200	53,200
Flip-Flops	28,800	68,800	81,200	35,200	92,400	106,400
Total Block RAM	1.8Mb	2.5Mb	3.8Mb	2.1Mb	3.3Mb	4.9Mb

Ans:也可以透過Zynq-7000 SoC Product Selection Guide查找得到。

2. 承上題，共有多少個RAMB36E1？

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	581	53200	1.09
LUTRAM	2	17400	0.01
FF	736	106400	0.69
BRAM	1	140	0.71
BUFG	1	32	3.13

Ans:PYNQ-Z2上總共有140顆BRAM。

2. 承上題，共有多少個RAMB36E1？

	Cost-Optimized Devices					
Device Name	Z-7007S	Z-7012S	Z-7014S	Z-7010	Z-7015	Z-7020
Part Number	XC7Z007S	XC7Z012S	XC7Z014S	XC7Z010	XC7Z015	XC7Z020
Processor Core	Single-Core ARM® Cortex™-A9 MPCore™ Up to 766MHz			Dual-Core ARM Cortex-A9 MPCore Up to 866MHz		
Processor Extensions	NEON™ SIMD Engine and Single/Double Precisi					
L1 Cache	32KB Instruction, 32KB Dat					
L2 Cache	512KB					
On-Chip Memory	256KB					
External Memory Support(2)	DDR3, DDR3L, DDR2					
External Static Memory Support(2)	2x Quad-SPI, NAN					
DMA Channels	8 (4 dedicated t					
Peripherals	2x UART, 2x CAN 2.0B, 2x I2C,					
Peripherals w/ built-in DMA(2)	2x USB 2.0 (OTG), 2x Tri-mode Giga					
Security(3)	RSA Authentication of First S AES and SHA 256b Decryption and Aut					
Processing System to Programmable Logic Interface Ports (Primary Interfaces & Interrupts Only)	2x AXI 32b Master, 2x A 4x AXI 64b/32b M AXI 64b ACI 16 Interrupt					
7 Series PL Equivalent	Artix®-7	Artix-7	Artix-7	Artix-7	Artix-7	Artix-7
Logic Cells	23K	55K	65K	28K	74K	85K
Look-Up Tables (LUTs)	14,400	34,400	40,600	17,600	46,200	53,200
Flip-Flops	28,800	68,800	81,200	35,200	92,400	106,400
Total Block RAM (# 36Kb Blocks)	1.8Mb (50)	2.5Mb (72)	3.8Mb (107)	2.1Mb (60)	3.3Mb (95)	4.9Mb (140)

Ans:也可以透過Zynq-7000 SoC Product Selection Guide查找得到。

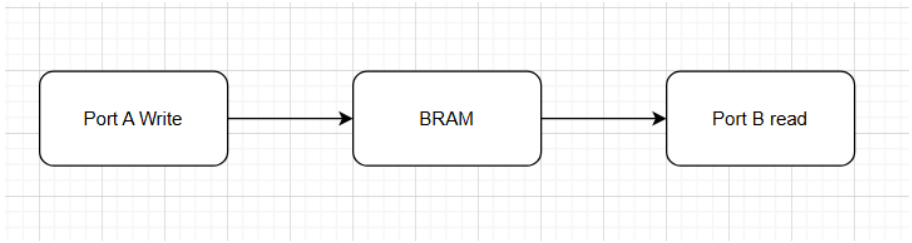
3. 若要將RAMB36E1 Configure成36Kb FIFO，該使用什麼Verilog Template？

```
module (
    input clk,
    input rst,
    input [31:0] data_in,
    input [3:0] w_en, // write enable
    input r_en, // read enable
    output reg empty,
    output reg full,
    output [31:0] data_out

);···

endmodule
```

實作32-bitwise的FIFO的I/O port



使用一顆TDP mode的BRAM，將要寫入FIFO的Data透過Port A寫入，要讀的Data透過Port B 讀出

3.若要將RAMB36E1 Configure成36Kb FIFO，該使用什麼Verilog Template？

```
reg [12:0] w_pointer;
reg [12:0] r_pointer;

// control w_pointer and r_pointer
always @(posedge clk or posedge rst) begin
    if (rst) begin
        w_pointer <= 13'd0;
        r_pointer <= 13'd0;
    end
    else begin
        w_pointer <= (w_en) ? w_pointer + 13'd1 : w_pointer;
        r_pointer <= (r_en) ? r_pointer + 13'd1 : r_pointer;
    end
end

// FIFO is empty cannot read
always @(*) begin
    empty = !(w_pointer[12] ^ r_pointer[12]) && (w_pointer[11:0] == r_pointer[11:0]);
    full = (w_pointer[12] ^ r_pointer[12]) && (w_pointer[11:0] == r_pointer[11:0]);
end
```

檢查 FIFO 的狀態，以w_pointer、r_pointer為FIFO的pointer，當兩pointer相等時，表示記憶體裡面為空。

Available Attributes

```
RAMB36E1 #(
  // Available Attributes
    .RAM_MODE("TDP"), // set as true dual port mode
    .DOA_REG(1), // set A port output register
    .DOB_REG(1), // set B port output register
    .READ_WIDTH_A(36), // A port read 32-bit data
    .WRITE_WIDTH_A(0), // A port cannot write
    .READ_WIDTH_B(0), // B port cannot read
    .WRITE_WIDTH_B(36) // B port write 32-bit data
)
```

RAMB36E1 在TDP mode操作，
使用 A port進行read操作、B port則為用來write。

Port Descriptions

```
.CLKARDCLK(clk), // A, B synchronize
.CLKBWRCLK(clk), // A, B synchronize
```

A B port 使用同一 clock 訊號

```
.REGCEAREGCE(1'b1), // enable A port output register
```

A port 作為 read port 所以設定為1

Port Descriptions

```
.ENARDEN(!empty), // read enable, cannot read when FIFO is empty

.ENBWREN(!full), // cannot write when FIFO is full
```

當FIFO empty 時 ，無法讀取 FIFO 內的內容
同理在 FIFO full 時，無法寫入FIFO。

Port Descriptions

```
.ADDRARDADDR({1'b0, w_pointer[11:0], 3'd0}), // read address
.ADDRBWRADDR({1'b0, r_pointer[11:0], 3'd0}), // write address
```

根據目前 write pointer 以及 read pointer 的位置來分別進行

- 使用 A port 讀取資料
- 使用 B port 寫入資料

Port Descriptions

```
.DOADO(data_out), // 32-bit data read out
.DIBDI(data_in), // 32-bit data write in
```

FIFO 的輸入與輸出資料

Verilog Template

```
RAMB36E1 #(
  // Available Attributes
    .RAM_MODE("TDP"), // set as true dual port mode
    .DOA_REG(1), // set A port output register
    .DOB_REG(1), // set B port output register
    .READ_WIDTH_A(36), // A port read 32-bit data
    .WRITE_WIDTH_A(0), // A port cannot write
    .READ_WIDTH_B(0), // B port cannot read
    .WRITE_WIDTH_B(36) // B port write 32-bit data
)

RAMB36E1_inst (
  // Port Descriptions

    .CLKARDCLK(clk), // A, B synchronize
    .CLKBWRCLK(clk), // A, B synchronize

    .REGCEAREGCE(1'b1), // enable A port output register
    .ENARDEN(!empty), // read enable, cannot read when FIFO is empty

    .ENBWREN(!full), // cannot write when FIFO is full
    .WEA(4'd0),// A port only for read
    .WEBWE({4'd0, w_en}),

    .ADDRARDADDR({1'b0, w_pointer[11:0], 3'd0}), // read address
    .ADDRBWRADDR({1'b0, r_pointer[11:0], 3'd0}), // write address

    .DOADO(data_out), // 32-bit data read out
    .DIBDI(data_in), // 32-bit data write in
);
```

完整設定