FPGA Homework5 第八組

F44071128 李其祐 C14074021 張柏彥 F14051041 陳 祺

Problem1: Simple Computing System

一、系統規格

BRAM0			
Signal Name	Direction	Width(bit)	Description
clk	Input	1	系統時脈訊號。
ADDRARDADDR	Input	12	PORT A 地址訊號
ENARDEN	Input	1	PORT A 致能訊號
WEA	Input	4	PORT A 寫入致能訊號
DIADI	Input	32	PORT A 寫入資料訊號
ADDRBWRADDR	Input	10	PORT B 讀取地址訊號
ENBWREN	Input	1	PORT B 致能訊號
DOBDO	Output	32	PORT B 輸出資料訊號

BRAM1			
Signal Name	Direction	Width(bit)	Description
clk	Input	1	系統時脈訊號。
ADADDR	Input	12	PORT A 讀取地址訊號
AWRADDR	Input	12	PORT A 寫入地址訊號
en_a	Input	1	PORT A 致能訊號
en_b	Input	1	PORT B 致能訊號
WEA	Input	4	PORT A 寫入致能訊號
WEBWE	Input	4	PORT B 寫入致能訊號
DIADI	Input	32	PORT A 寫入資料訊號
DIBDI	Input	48	PORT B 寫入資料訊號
BWRADDR	Input	10	PORT B 寫入地址訊號
BRADDR	Input	10	PORT B 讀取地址訊號
DOADO	Output	32	PORT A 輸出資料訊號
DOBDO	Output	32	PORT B 輸出資料訊號

DSP			
Signal Name	Direction	Width(bit)	Description
clk	Input	1	系統時脈訊號。

A	Input	32	A 的輸入資料訊號
B	Input	32	B 的輸入資料訊號
INMODE	Input	5	控制 DSP48E1 Slice 中選擇預加器、A、B 和 D 輸入的訊號
ALUMODE	Input	4	控制 DSP48E1 Slice 中運算功能的訊號
OPMODE	Input	7	控制 DSP48E1 Slice 中 X、Y 和 Z 多工器的輸入的訊號
P	Output	48	輸出資料訊號

Controller			
Signal Name	Direction	Width(bit)	Description
clk	Input	32	系統時脈訊號。
rst_n	Input	1	同步 active low 重置訊號。
en	Input	1	資料有效輸入訊號。當此訊號為 1 時表示輸入資料有效。
inst	Input	32	指令訊號，其中 inst[31]為 Execute ; inst[30:27]為 DSP_ALUMODE ; inst[26:20]為 DSP_OPMODE ; inst[19:15]為 DSP_INMODE ; inst[14:10]為 BRAM1_Write_Addr ; inst[9:5]為 BRAM1_Read_Addr ; inst[4:0]為 BRAM0_Read_Addr。
valid	Output	1	資料寫入完成訊號。當此訊號為 1 時表示經由 DSP 運算完的值已寫入 BRAM1。
bram0_raddr	Output reg	10	BRAM0 的讀取地址，為 inst[4:0]。
bram0_enb	Output reg	1	BRAM0 PORT B 致能訊號
bram1_addrb	Output reg	10	BRAM1 的地址，當 cs==READ 時為 {5'b0, inst[9:5]}，當 cs==WRITE 時為 {5'b0, inst[14:10]}。
bram1_web	Output reg	4	PORT B 寫入致能訊號
bram1_enb	Output reg	1	BRAM1 PORT B 致能訊號
dsp_alumode	Output reg	4	控制 DSP48E1 Slice 中運算功能的訊號
dsp_opmode	Output reg	7	控制 DSP48E1 Slice 中 X、Y 和 Z 多工器的輸入的訊號
dsp_inmode	Output reg	5	控制 DSP48E1 Slice 中選擇預加器、A、B 和 D 輸入的訊號
inst_reg	reg	31	用於暫存 inst[30:0]

cs	reg	3	暫存 FSM 目前的狀態
ns	reg	3	暫存 FSM 下一個狀態
cnt	reg	2	時長為 3 個 cycle 的計數器，用於讓 cs 維持在 EXE 狀態。

二、電路設計說明

1. BRAM0

Data width: 32bit

將 Attributes 中 READ_WIDTH/WRITE_WIDTH 設為 36

```
// 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
```

RAM Mode: TDP

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

Initial Contents:

```
.INIT_00(256'h0000000000005310000000000000000000000000120100000000100000023),
.INIT_01(256'h00000000000000000000000000000FFFFFF230000000000000000000000000),
.INIT_02(256'h00000000000000000000000000000000000000000000000000000000000000),
```

Port A: BRAM0 Port A (Write Only)連接 AXI BRAM Controller，因此將 port A 的 Address/Controls Input Signal 與 Data Input Signal 依下圖設定:

(1) Address/Controls Input Signal

```
// Port A Address/Control Signals: 16-bit (each) input: Port A address and control signals (read port
// when RAM_MODE="SDP")
.ADDRARDADDR({1'b0, ADDRARDADDR[11:0], 3'b000}), // 16-bit input: A port address/Read address
.CLKARDCLK(clk), // 1-bit input: A port clock/Read clock
.ENARDEN(ENARDEN), // 1-bit input: A port enable/Read enable
.REGCEAREGCE(ENARDEN), // 1-bit input: A port register enable/Register enable
.RSTRAMARSTRAM(), // 1-bit input: A port set/reset
.RSTREGARSTREG(), // 1-bit input: A port register set/reset
.WEA(WEA), // 4-bit input: A port write enable
```

2. Data Input Signal

```
// Port A Data: 32-bit (each) input: Port A data
.DIADI(DIADI), // 32-bit input: A port data/LSB data
.DIPADIP(), // 4-bit input: A port parity/LSB parity
```

Port B: BRAM0 Port B (Read Only)的輸入連接 Controller，因此將 port B 的 Address/Controls Input Signal 依下圖設定。

```
// Port B Address/Control Signals: 16-bit (each) input: Port B address and control signals (write port
// when RAM_MODE="SDP")
.ADDRBWRADDR({1'b0, ADDRBWRADDR[9:0], 5'b00000}), // 16-bit input: B port address/Write address
.CLKBWRCLK(clk), // 1-bit input: B port clock/Write clock
.ENBWREN(ENBWREN), // 1-bit input: B port enable/Write enable
.REGCEB(ENBWREN), // 1-bit input: B port register enable
.RSTRAMB(), // 1-bit input: B port set/reset
.RSTREGB(), // 1-bit input: B port register set/reset
.WEBWE(), // 8-bit input: B port write enable/Write enable
```

而 BRAM0 Port B 的輸出連接 DSP，因此將 port B 的 Data Output Signal 依下圖設定。

```
// Port B Data: 32-bit (each) output: Port B data
.DOBDO(DOBDO), // 32-bit output: B port data/MSB data
```

2. BRAM1

Data width: 32bit

將 Attributes 中 READ_WIDTH/WRITE_WIDTH 設為 36

```
// 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
```

RAM Mode: TDP

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

Initial Contents:

```
.INIT_00(256'h0000000000005310000000000000000000000000120100000001000000023),
.INIT_01(256'h00000000000000000000000000000FFFFFF2300000000000000000000000000),
.INIT_02(256'h00000000000000000000000000000000000000000000000000000000000000),
```

BRAM1 PortA (Read / Write) 輸入與輸出連接 AXI_BRAM_Controller，因此 Port A Data Output Signal、Port A Data Input Signal、Port A Address/Control Input Signal 依下圖設定。

(1) Port A Data Output:

```
// Port A Data: 32-bit (each) output: Port A data
.DOADO(DOADO), // 32-bit output: A port data/LSB data
.DOPADOP(), // 4-bit output: A port parity/LSB parity
```

(2) Port A Data Input:

```
// Port A Data: 32-bit (each) input: Port A data
.DIADI(DIADI), // 32-bit input: A port data/LSB data
.DIPADIP(), // 4-bit input: A port parity/LSB parity
```

(3) Port A Address/Control Input:

```
// Port A Address/Control Signals: 16-bit (each) input: Port A address and control signals (read port
// when RAM_MODE="SDP")
.ADDRARDADDR({1'b0, ADDRARDADDR[11:0], 3'b000}), // 16-bit input: A port address
.CLKARDCLK(clk), // 1-bit input: A port clock/Read clock
.ENARDEN(en_a), // 1-bit input: A port enable
.REGCEAREGCE(en_a), // 1-bit input: A port register enable/Register enable
.RSTRAMARSTRAM(), // 1-bit input: A port set/reset
.RSTREGARSTREG(), // 1-bit input: A port register set/reset
.WEA(WEA), // 4-bit input: A port write enable
```

由於 Port B 的輸入可能來自 Controller 或 DSP，因此用 en_b(port B enable)與 WEBWE(port B write enable)控制其 address。若 en_b 為 1 且 WEBWE=f 就代表此時在寫入狀態，因此要傳送寫入地址，反之則為讀取狀態，因此要設定讀取地址。

```
wire [9:0]  ADDRBWRADDR = (en_b) ? ((WEBWE == 'hf) ? BWRADDR : BRADDR) : 0;
```

而 Port B Data Input Signal、Port B Address/Control Input Signal 依下圖設定。

(1) Port B Input:

```
// Port B Data: 32-bit (each) input: Port B data
.DIBDI(DIBDI[31:0]), // 32-bit input: B port data/MSB data
.DIPBDIP() // 4-bit input: B port parity/MSB parity
```

(2) Port B Address/Control Input:

```
// Port B Address/Control Signals: 16-bit (each) input: Port B address and control signals (write port
// when RAM_MODE="SDP")
.ADDRBWRADDR({1'b0, ADDRBWRADDR[9:0], 5'b00000}), // 16-bit input: B port address
.CLKBWRCLK(clk), // 1-bit input: B port clock/Write clock
.ENBWREN(en_b), // 1-bit input: B port enable
.REGCEB(en_b), // 1-bit input: B port register enable
.RSTRAMB(), // 1-bit input: B port set/reset
.RSTREGB(), // 1-bit input: B port register set/reset
.WEBWE({4'b0000,WEBWE}), // 8-bit input: B port write enable/Write enable
```

(3) Port B 的輸出連接 DSP，因此 Port B Data Output Signal 依下圖設定。

```
// Port B Data: 32-bit (each) output: Port B data
.DOBDO(DOBDO), // 32-bit output: B port data/MSB data
.DOPBDOP(), // 4-bit output: B port parity/MSB parity
```

3. DSP:

(1) Feature Control Attributes : 將 USE_DPORT 設定為"FALSE"

```
// Feature Control Attributes: Data Path Selection
.A_INPUT("DIRECT"), // Selects A input source, "DIRECT" (A port) or "CASCADE" (ACIN port)
.B_INPUT("DIRECT"), // Selects B input source, "DIRECT" (B port) or "CASCADE" (BCIN port)
.USE_DPORT("FALSE"), // Select D port usage (TRUE or FALSE)
.USE_MULT("MULTIPLY"), // Select multiplier usage ("MULTIPLY", "DYNAMIC", or "NONE")
.USE_SIMD("ONE48"), // SIMD selection ("ONE48", "TWO24", "FOUR12")
```

(2) Register Control Attributes :

將 Attributes 中 AREG、BREG、PREG 的數目設成 1。

```
// Register Control Attributes: Pipeline Register Configuration
.ACASCREG(1), // Number of pipeline stages between A/ACIN and ACOUT (0, 1 or 2)
.ADREG(0), // Number of pipeline stages for pre-adder (0 or 1)
.ALUMODEREG(1), // Number of pipeline stages for ALUMODE (0 or 1)
.AREG(1), // Number of pipeline stages for A (0, 1 or 2)
.BCASCREG(1), // Number of pipeline stages between B/BCIN and BCOUT (0, 1 or 2)
.BREG(1), // Number of pipeline stages for B (0, 1 or 2)
.CARRYINREG(0), // Number of pipeline stages for CARRYIN (0 or 1)
.CARRYINSELREG(0), // Number of pipeline stages for CARRYINSEL (0 or 1)
.CREG(0), // Number of pipeline stages for C (0 or 1)
.DREG(0), // Number of pipeline stages for D (0 or 1)
.INMODEREG(1), // Number of pipeline stages for INMODE (0 or 1)
.MREG(1), // Number of multiplier pipeline stages (0 or 1)
.OPMODEREG(1), // Number of pipeline stages for OPMODE (0 or 1)
.PREG(1) // Number of pipeline stages for P (0 or 1)
```

(要注意的是若 A(B)REG 設定為 1, 則 A(B)CASCREG U 要設定為 1)

(3) Control Input：將 CARRYINSEL 設成 0，其他的控制訊號連接依下圖設定。

```
// Control: 4-bit (each) input: Control Inputs/Status Bits
.ALUMODE(ALUMODE), // 4-bit input: ALU control input
.CARRYINSEL(0), // 3-bit input: Carry select input
.CLK(clk), // 1-bit input: Clock input
.INMODE(INMODE), // 5-bit input: INMODE control input
.OPMODE(OPMODE), // 7-bit input: Operation mode input
```

(4) Data Input：取 A[29:0]與 B[17:0]用於乘法運算(符合 LSB)，Port C 固定為常數 0x0000_0009_5514，並設定 CARRYIN 為 0。

```
// Data: 30-bit (each) input: Data Ports
.A(A[29:0]), // 30-bit input: A data input
.B(B[17:0]), // 18-bit input: B data input
.C(48'h000000095514), // 48-bit input: C data input
.CARRYIN(0), // 1-bit input: Carry input signal
.D(), // 25-bit input: D data input
```

4. Controller

將 state 分為 5 個 state，分別為 IDLE、READ、EXE、WRITE、DONE。

一開始在 IDLE 時，當 en 為 1 且 inst[31]為 1 時，next state 會變成 READ，而若 en 為 0，就維持在 IDLE，若 inst[31]=0，則直接跳至 DONE。

當在 READ state 時，會將 inst_reg(也就是存放 inst 的暫存器) 依照作業規定進行 decode，因此將 inst_reg[4:0], inst_reg[9:5]的值 assign 給 bram0 和 bram1 的 read address，而為了配合 bram 的設定，就會將缺少的 bit 數補 0。此外，由於要使用到兩個 bram 的 port B，並且為 read enable，因此要設定 bram1_enb=1 且 bram1_web=0 以及 bram0_enb=1，bram0/bram1 就會將對應 address 的直送入 DSP 來進行運算。Next state 設為 EXE。

當讀完 DSP 所需資料就跳入 EXE state，此 state 一樣也依照作業規定進行 decode，將 inst_reg[19:15], inst_reg[26:20], inst_reg[30:27]的值分別指派給 dsp_inmode, dsp_opmode, dsp_alumode。此外，此 state 的另一個目的是在等待 DSP 計算完成，因此我們透過 cnt 來計數，若 cnt 數到設定的 3 個 cycle (DSP 正確輸出結果會是 3 個 cycle 後)，才進入下一個 WRITE state。

當在 WRITE state 時，一樣依照作業規定進行 decode，因此將 inst_reg[14:10]的值指派給 bram1_addrb，這裡一樣要使用到 bram1 的 port B 並且為 write enable，因此設定 bram1_enb=1 且 bram1_web=4'hf (這裡設定 f 是因為總共為 4byte, 32bit 的資料)，bram1 就會將 DSP 運算後的值寫入至對應的 address。Next state 設為 DONE。

當在 DONE state 時，將 valid 拉為 1，表示已經將正確的運算結果寫入 BRAM1。而 next state 的設定則會根據 en 是否為 0，來決定是否跳到 IDLE state 來進行下一筆資料的運算，會這樣設計是為了避免 FSM 和預期的不同，提早開始作動。

三、main.c 程式說明

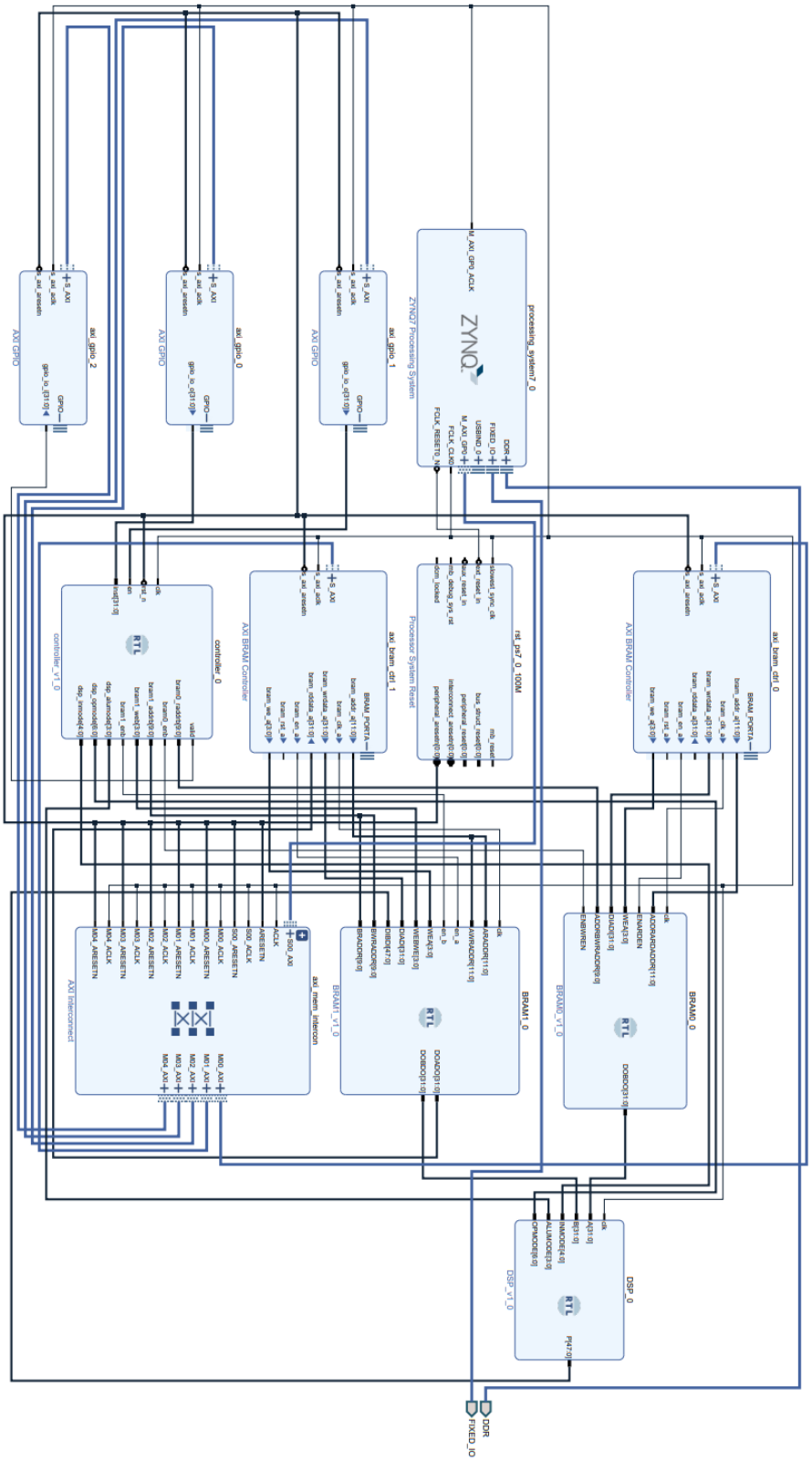
1. get_inst(bram0_raddr, bram1_raddr, bram1_waddr, opmode_Z, alumode):
此函式目的在於將 address 和 dsp mode 資訊存入 instrustion，因此就將 bram0_raddr, bram1_raddr, bram1_waddr, inmode, opmode, alumode 依據作業規定存到 inst，其中 inmode 不作為參數是因為此次作業中皆選擇 A1(即 inmode[3:0]=4'b0001)、B1(inmode[4]=1'b1)為輸入，因此 inst += 0b10001 <<15。而 opmode 只需將 z 的資訊作為參數即可，因為在此次作業中 X、Y 多工器皆需輸出 M，因此 opmode[1:0]與 opmode[3:2]皆為 2'b01(即 X=Y=2'b01)，因此 inst += 0b0101 << 20 ; 而若 Z 當作 0 時，則 opmode[6:4]設為 0 ;若 Z 當作 C 時，則 opmode[6:4]設為 3'b011。alumode 為 inst[30:27]，在此次作業中，當執行的運算為 A*B+C 時，alumode 為 4'b0000、當執行的運算為 C-A*B 時，alumode 為 4'b0011、當執行的運算為 A*B-C-1 時，alumode 為 4'b0001，因此 inst += alumdoe << 27。Execute 訊號為 inst[31]，當需要執行指令時，其值為 1，此值是用來驅動 controller.v 的 FSM。

2. exec(inst):

此函式的目的在於精簡整個流程，每次要執行一次題目所規定的運算都必須重複以下步驟: 送指令，送 en=1 訊號，等待 valid=1，送 en=0 訊號。

由下圖的 block design 可得知我們分別透過 GPIO0, GPIO1, GPIO2 來送 inst, en 訊號給 controller 和接受 controller 的 valid 訊號，因此呼叫 Xil_Out32(XPAR_AXI_GPIO_0_BASEADDR, inst)來將 inst 送入 PL 端，和呼叫 Xil_Out32(XPAR_AXI_GPIO_1_BASEADDR, 1)來將 enable = 1 的訊號送入 PL 端，這樣一來整個運算系統就會被驅動。當 DSP 運算完後，要寫入時，會把 valid 訊號拉高，其他時候都是 0，因此就可以透過 while loop 來偵測現在 valid 是否為 0，若為 0 就一直卡在 while，當 valid 回傳 1 時，會再透過 GPIO1 送 en=0 的訊號給 controller，使 state 回到 IDLE。

四、Block Design



五、功能驗證

1. Step1

COM4 - PuTTY

```
HW 5-1 Program en.

Step1
BRAM1[0] = 0x23
BRAM1[1] = 0x1
BRAM1[2] = 0x1201
BRAM1[3] = 0x27623
BRAM1[4] = 0x0
BRAM1[5] = 0x0
BRAM1[6] = 0x531
BRAM1[7] = 0x15403c9
BRAM1[8] = 0x0
BRAM1[9] = 0x0
BRAM1[10] = 0x8ad37a
BRAM1[11] = 0xffffff23
BRAM1[12] = 0x0
BRAM1[13] = 0x94fe3
BRAM1[14] = 0x0
BRAM1[15] = 0xfffb584d
BRAM1[16] = 0x0
BRAM1[17] = 0x0
BRAM1[18] = 0x0
BRAM1[19] = 0x0
BRAM1[20] = 0x0
BRAM1[21] = 0x0
BRAM1[22] = 0x0
BRAM1[23] = 0x0
BRAM1[24] = 0x0
BRAM1[25] = 0x0
BRAM1[26] = 0x0
BRAM1[27] = 0x0
BRAM1[28] = 0x0
BRAM1[29] = 0x0
BRAM1[30] = 0x0
BRAM1[31] = 0x2236
```

以第一題為例，將 BRAM0[0]*BRAM1[2]的結果寫入 BRAM[3]，因此設定參數 bram0_raddr=0, bram1_raddr=2, bram1_waddr=3, opmode_Z=0b000, alumode=0b0000，將得到的 inst 傳入 exec 函式後，經由 GPIO0 送入至 PL 端來進行運算。0x23*0x1201=0x27623，與結果相符。

2. Step2:

```
Step2
BRAM1[0] = 0x23
BRAM1[1] = 0x1
BRAM1[2] = 0x1201
BRAM1[3] = 0x27623
BRAM1[4] = 0x0
BRAM1[5] = 0x0
BRAM1[6] = 0x531
BRAM1[7] = 0x15403c9
BRAM1[8] = 0x0
BRAM1[9] = 0x0
BRAM1[10] = 0x8ad37a
BRAM1[11] = 0xffffff23
BRAM1[12] = 0x0
BRAM1[13] = 0x94fe3
BRAM1[14] = 0x0
BRAM1[15] = 0xfffb584d
BRAM1[16] = 0x1201
BRAM1[17] = 0xff2273b0
BRAM1[18] = 0x187914
BRAM1[19] = 0x94050
BRAM1[20] = 0xfff6cd21
BRAM1[21] = 0x0
BRAM1[22] = 0x0
BRAM1[23] = 0x0
BRAM1[24] = 0x0
BRAM1[25] = 0x0
BRAM1[26] = 0x0
BRAM1[27] = 0x0
BRAM1[28] = 0x0
BRAM1[29] = 0x0
BRAM1[30] = 0x0
BRAM1[31] = 0x2236

HW 5-1 Program Done.
```

以第一題為例，將 BRAM0[0]*BRAM1[2]的結果寫入 BRAM[16]，因此設定參數 bram0_raddr=0, bram1_raddr=2, bram1_waddr=16, opmode_Z=0b000, alumode=0b0000，將得到的 inst 傳入 exec 函式後，經由 GPIO0 傳送進行運算。0x1*0x1201=0x1201，與結果相符。

將 step1 和 step2 的結果與助教提供的截圖比對，可以發現都是正確的。

Problem

1. PYNQ-Z2 上共有多少個 DSP48E1 slice?

ANS: 根據其 user manual (reference: pynqz2_user_manual_v1_0-1525725.pdf (mouser.com))，可得知總共有 220 個 DSP48E1 slice

1 PYNQ-Z2 features

- **ZYNQ XC7Z020-1CLG400C**
  - 650MHz ARM® Cortex®-A9 dual-core processor
  - Programmable logic
    - 13,300 logic slices, each with four 6-input LUTs and 8 flip-flops
    - 630 KB block RAM
    - 220 DSP slices
    - On-chip Xilinx analog-to-digital converter (XADC)
  - Programmable from JTAG, Quad-SPI flash, and MicroSD card
- **Memory and storage**
  - 512MB DDR3 with 16-bit bus @ 1050Mbps
  - 16MB Quad-SPI Flash with factory programmed 48-bit globally unique EUI-48/64™ compatible identifier
  - MicroSD slot