

FPGA Homework3 第八組

F44071128 李其祐 C14074021 張柏彥 F14051041 陳祺

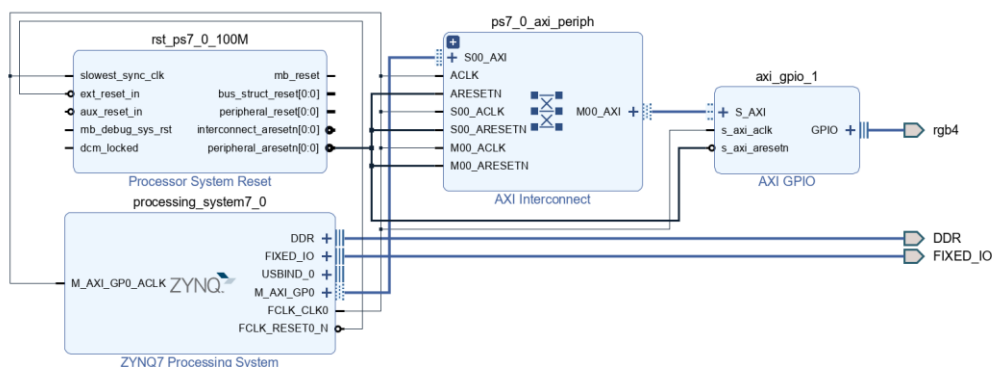
Problem1: RGB Light

一、led.c 程式說明

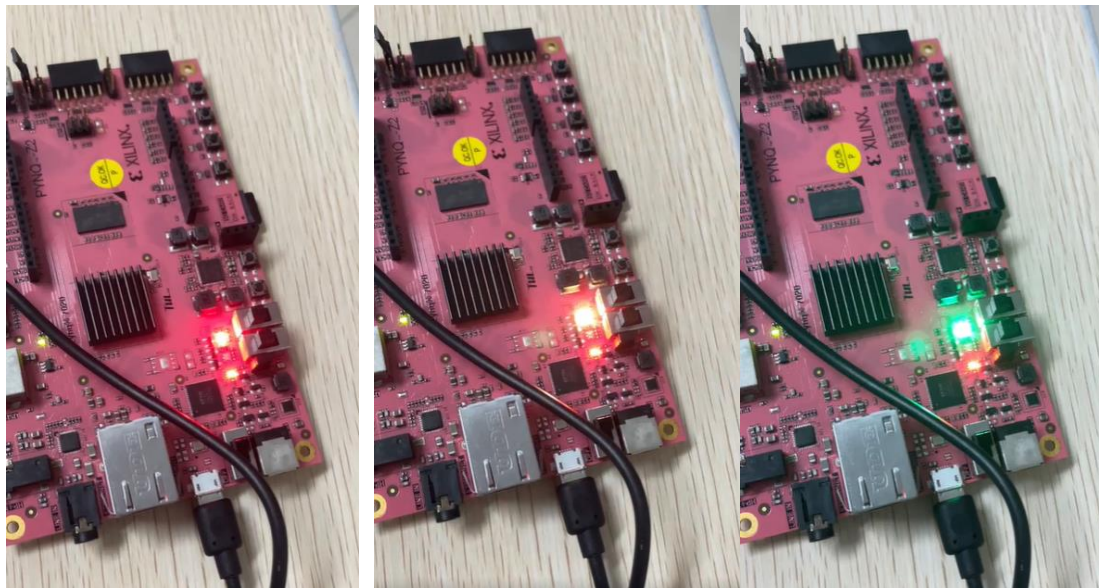
首先將 LED_DELAY 定義為 2560000 個 cycle，並宣告 XGpio LED_Gpio，接著在 main 函式裡執行 XGpio_SetDataDirection(&LED_Gpio, 1, 0x00)，設定 LED_Gpio 為輸出且通道為 1。

接著在 while 迴圈裡執行 `XGpio_DiscreteWrite(&LED_Gpio, 1, 0x00)`，將 RGB LED 設定為 High。接著執行三個 for 迴圈，其中 i 表示有 6 個 state，根據不同的 state 控制 RGB 的輸出；而 Delay 則是控制等待時長，拉長 RGB LED 的發光時間；最後利用 counter 以 256 個 cycle 為一時間單位。當 i=0 時，執行 `Gpio_DiscreteWrite(&LED_Gpio, 1, 0b001)`使 RGB LED 亮紅光；當 i=1 時，當 counter<=97 時執行 `Gpio_DiscreteWrite(&LED_Gpio, 1, 0b011)`使 RGB LED 亮綠光與紅光，當 counter>97 時執行 `Gpio_DiscreteWrite(&LED_Gpio, 1, 0b001)`使 RGB LED 只亮紅光；當 i=2 時，執行 `Gpio_DiscreteWrite(&LED_Gpio, 1, 0b011)`使 RGB LED 亮綠光與紅光；當 i=3 時，執行 `Gpio_DiscreteWrite(&LED_Gpio, 1, 0b010)`使 RGB LED 亮綠光；當 i=4 時，執行 `Gpio_DiscreteWrite(&LED_Gpio, 1, 0b100)`使 RGB LED 亮藍光。當 i=5 時，當 counter<=31 時執行 `Gpio_DiscreteWrite(&LED_Gpio, 1, 0b111)`使 RGB LED 亮藍光綠光與紅光，當 counter<=127 時執行 `Gpio_DiscreteWrite(&LED_Gpio, 1, 0b101)`使 RGB LED 亮藍光與紅光，當 counter>97 時執行 `Gpio_DiscreteWrite(&LED_Gpio, 1, 0b100)`使 RGB LED 只亮藍光。最後執行 `XGpio_DiscreteClear(&LED_Gpio, 1, 0x00)`使 RGB LED 停止發光。

二、Block Design



三、FPGA 驗證



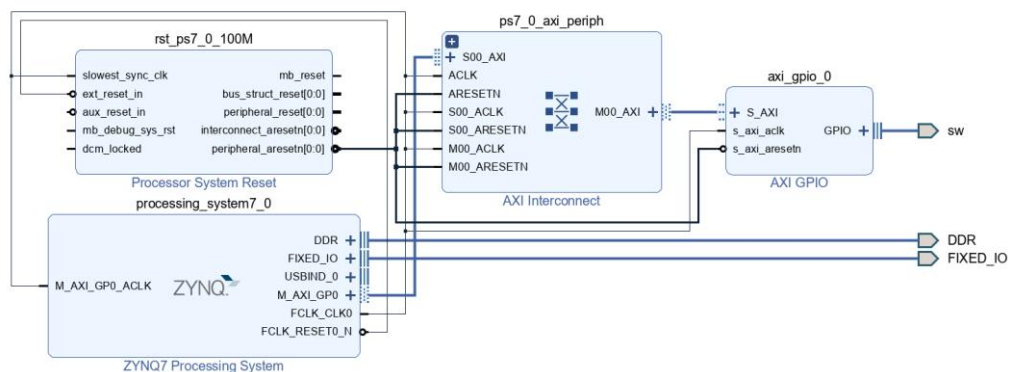
Problem2: Sorting

一、sort.c 程式說明

首先宣告 XGpio SW_Gpio，接著在 main 函式裡執行 XGpio_SetDataDirection(&SW_Gpio, 1, 0x0f)，設定 SW_Gpio 為輸入且通道為 1，並執行 XGpio_DiscreteRead(&SW_Gpio, 1) 讀取 Switch 的值到 sw_data。接著宣告一個大小為 20 的 int 陣列 num，在 for 迴圈執行 20 次 scanf 在 PUTTY 上依序輸入數字到 num 陣列。經過 bubble_sort 函式排列後，再用 for 迴圈將排序後的結果顯示在 PUTTY 上。

本次作業使用的是 bubble sort，bubble sort 的原理是將兩個相鄰的數值相比，若要升序排列，則如果前一個數值比後一個數值大時，就互相對調；反之，若要降序排列，則如果前一個數值比後一個數值小時，互相對調。此外根據這次作業的要求，將由 sw_data 的值決定為升序或是降序排列。

二、Block Design



三、FPGA 驗證

COM4 - PuTTY

switches data = 1
Please enter 20 non-negative integers:
1
4
2
17
13
9
43
27
76
39
23
62
54
81
83
33
45
97
47
59
After Sorting:
97 83 81 76 62 59 54 47 45 43 39 33 27 23 17 13 9 4 2 1
Successfully ran Gpio Example

COM4 - PuTTY

switches data = 0
Please enter 20 non-negative integers:
1
4
2
17
13
9
43
27
76
39
23
62
54
81
83
33
45
97
47
59
After Sorting:
1 2 4 9 13 17 23 27 33 39 43 45 47 54 59 62 76 81 83 97
Successfully ran Gpio Example

Problem3

一、系統規格

Arithmetic			
Signal Name	Direction / Type	Width(bit)	Description
din1	Input signed	8	第一個運算元
din2	Input signed	8	第二個運算元
op	Input	2	代表運算子，其中 2'b00 代表 '+'、2'b01 代表 '-'、2'b10 代表 '*'
sign_ed	Output	1	代表符號位，1 代表運算後的值為負數，0 代表運算後的值為正數
dout	Output	7	運算後結果的數值
overflow	Output reg	1	檢查有無發生溢位，1 代表有溢位發生，0 則無
result	reg signed	32	運算後的結果

Sorting			
Signal Name	Direction / Type	Width(bit)	Description
din	Input	32	由 8 個欲排序的數組成，其中 din[3:0] 代表第一個數、din[7:4] 代表第二個數、....、din[31:28] 代表第八個數。
dout	Output reg	32	由 8 個已排序的數組成，其中 dout[3:0] 代表第一個數、dout[7:4] 代表第二個數、....、dout[31:28] 代表第八個數
temp	reg	4	兩數在交換時，用於暫存其中一個數值
array	reg	4*8	用於暫存所有欲排序的數值的陣列

Parity Generator			
Signal Name	Direction	Width(bit)	Description
in	Input	32	輸入的 32bit 資料
Parity	Output	1	代表 parity bit，其中 1 代表輸入的數值在二進位表示下有奇數個 1，0 則代表輸入的數值在二進位表示下有偶數個 1

二、電路設計說明

1. Arithmetic:

首先根據輸入的 op，分成不同的情況。若 op 為 0，代表要執行加法，因此 $result = din1 + din2$ ，overflow 則判斷是否有兩正數相加卻出現負數的情況；或是兩負數相加出現正數的情況，若有則 overflow 拉為 1；若 op 為 1，代表要執行減法，因此 $result = din1 - din2$ ，overflow 則判斷兩種情況，正數-負數=負數以及負數-正數=正數，若發生此兩種情況 overflow 就拉為 1；若 op 為 2，代表要執行乘法，因此 $result = din1 * din2$ ，要注意的是乘法的 overflow 必須根據 result[31] 判斷，若 result[31] = 1，代表運算結果為負數，因此若 result[7:0] > 128 時才會發生 overflow，反之若 result[31] = 0，代表運算結果為正數，因此若 result[7:0] > 127 時才會發生 overflow。最後輸出部分將 result[6:0] 傳給 dout，result[31] 傳給 sign_ed。

2. Sorting:

首先先將 `din` 代表的 8 個數字寫入 `array`，同樣使用 `bubble sort` 的方式，將兩個相鄰的數值相比，如果前一個數比後一個數小，則互相對調，因此運算結束後會在 `array` 得到一個降序的排列。最後再將 `{array[7], array[6], ..., array[0]}` 傳給 `dout`。

3. Parity Generator:

由於 `parity` 是用來判斷輸入的數在二進位表示下 1 的個數是奇數還是偶數，於是我們將輸入值做 `bitwise-XOR`，因此若有奇數個 1，`parity` 則為 1；若有偶數個 1，`parity` 則為 0。

三、C 程式說明

1. Arithmetic (`arith()`)

首先，由於輸入的兩個數值 `A`、`B` 的資料型態為 `u32`，而 `A`、`B` 有可能為負數導致 8~31bit 因為 `sign extension` 的原因變為 1，因此我們需要將 `A`、`B` 先 `&0b11111111` 以確保 8~31bit 的數值為 0，才能將 `A+(B << 8)+(op << 16)` 的值傳給 `data`，再透過執行定義的 `ARITH_mWriteReg(baseAddr, 0, data)`，將 `data` 寫入 `reg0`。而經過運算後的數值就透過 `r_data` 寫入 `reg1`，接著我們就可以透過 `ARITH_mReadReg` 得到 `*of`、`sign`、`value` 以及 `result`，其中 `*of` 代表 `overflow`、`sign` 代表符號位元、`value` 代表運算後結果的數值(不包含正負號)，`result` 則為運算後的值，最後的回傳值會先判斷正負號，還有 `value` 是否為 0，來決定是否要將 `result+1`。

2. Sorting (`sort()`)

透過執行定義好的 `Sort_mWriteReg(baseAddr, 0, in)`，將輸入的值 `in` 寫入 `reg0`。再將運算好的值透過定義好的 `Sort_mReadReg(baseAddr, 8)` 從 `reg2` 中取出並回傳其值。

3. Parity Generator (`pg()`)

透過執行定義好的 `PG_mWriteReg(baseAddr, 0, A)`，將輸入的值 `A` 寫入 `reg0`。再將運算好的值透過定義好的 `PG_mReadReg(baseAddr, 12)` 從 `reg3` 中取出並回傳其值。

4. Main

一開始會先請使用者輸入模式，其中 1 代表 `Arithmetic mode`；2 代表 `Sorting mode`；3 代表 `Parity Generator mode`，以下說明各模式的細節。

若輸入 1，則執行 `Arithmetic`，首先會先讓使用者依序輸入第一個運算子 `A`、第二個運算子 `B` 與運算元 `operator`，若運算元為 '+'，則 `op=0`，若運算元

為'-'，則 **op=1**，若運算元為'*'，則 **op=2**，接著宣告代表是否 **overflow** 的變數 **of**，並將 **A**，**B**，**op** 與 **of** 傳入 **arith** 函式，最後就能透過 **arith** 函式得回傳值得到運算結果與 **overflow** 與否。


若輸入 **2**，則執行 **Sorting**，一開始會先請使用者輸入 **8** 個數字(一行一個)，這裡透過 **for** 迴圈來完成，並在每次的迴圈中將使用者的輸入值 **in**，累加至 **arr** 中，總共執行 **8** 次。接著就將 **arr** 傳入 **sort** 函式，並且得到 **sort** 函式的回傳排序後的結果 **aftersort**。最後就透過 **for** 迴圈來輸出排序後的數列(一次輸出 **4bit**)。

若輸入 **3**，則執行 **Parity Generator**，一開始會先請使用者在輸入十進位整數，接著就將輸入值 **din** 傳入 **pg** 函式，最後此函式就會回傳 **parity bit**。

此外，當每個模式執行完後就會跳回主選單，讓使用者重新選擇模式。

四、功能驗證

1. Arithmetic

 COM4 - PuTTY

```
Program Start.
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 1
Input A:64
Input B:63
Input Operator:+
Result = 127
Overflow = 0
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 1
Input A:127
Input B:1
Input Operator:+
Result = 0
Overflow = 1
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 1
Input A:-64
Input B:64
Input Operator:-
Result = -128
Overflow = 0
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 1
Input A:-128
Input B:1
Input Operator:-
Result = 0
Overflow = 1
```

```

Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 1
Input A:63
Input B:2
Input Operator:*
Result = 126
Overflow = 0
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 1
Input A:64
Input B:2
Input Operator:*
Result = 0
Overflow = 1
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 1
Input A:-64
Input B:-2
Input Operator:*
Result = 0
Overflow = 1
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 1
Input A:64
Input B:-2
Input Operator:*
Result = -128
Overflow = 0

```

2. Sorting

COM4 - PuTTY

```

Program Start.
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 2
Please input 8 elements array (Enter one element in a row):
15
13
12
14
1
5
8
2
15 14 13 12 8 5 2 1

```

3. Parity generator

COM4 - PuTTY

```

Program Start.
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 3
Input data:127
Parity bit = 1
Please input a mode (1. arithmetic, 2.sorting, 3.parity generator, 4. Exit): 3
Input data:255
Parity bit = 0

```

五、Block Design

