411440117

數位組 特殊招生作業

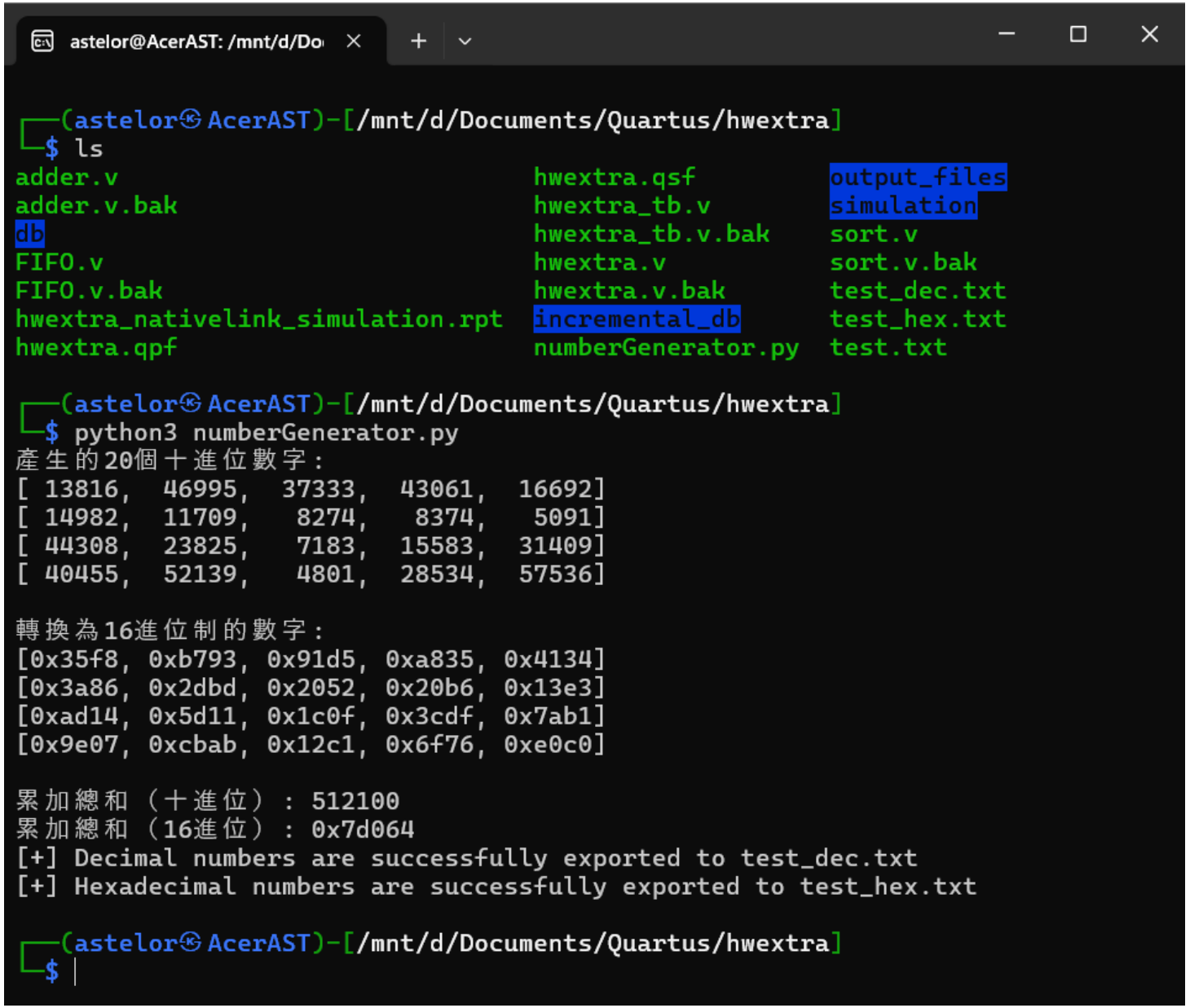
數字生成 1/3

- numberGenerator.py是作業隨附的範例腳本，再加上把生成出的數字輸出到文字檔的功能 (如圖)。
- 我相信應該能用tcl之類的Quartus腳本語言，將人工使用terminal自行跑python的步驟省去，自動化數字生成，但是礙於時間限制，這部分有待研究。

```
# output the generated numbers to a text file
print("[+] Decimal numbers are successfully exported to test_dec.txt")
with open("test_dec.txt", "w") as text_file:
    for i in numbers:
        print(f"{i}",file = text_file)


print("[+] Hexadecimal numbers are successfully exported to test_hex.txt")
with open("test_hex.txt", "w") as text_file:
    for i in hex_numbers:
        print(f"{i}",file = text_file)
~
~
~
"numberGenerator.py" [readonly][dos] 53L, 1749B
```

數字生成 2/3

- 使用Linux terminal和python3跑numberGenerator.py的結果(如圖)。
- (我只會用Linux terminal，而且Linux terminal比較好用)
- ModelSim每進行一次模擬會再讀一次程式內指定的文字檔，若數字重新生成，只需要用ModelSim中的Restart重跑一次模擬即可。



數字生成 3/3

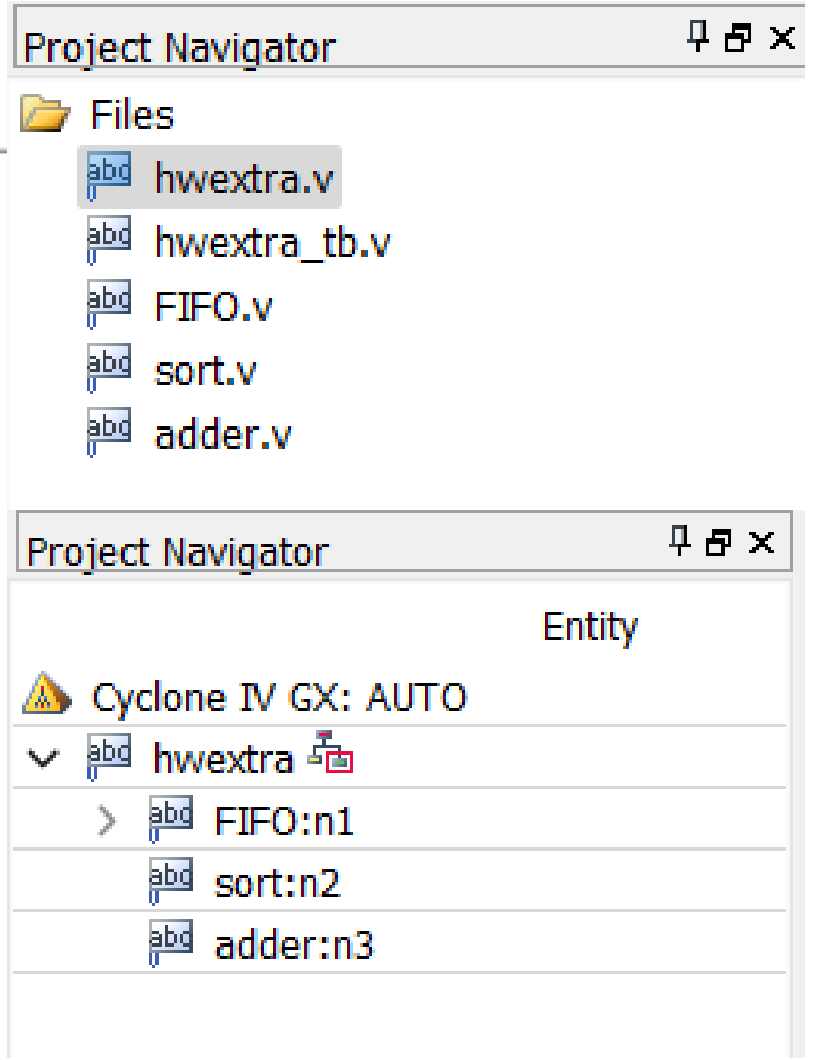
- 輸出文字檔(如圖)

```
┌──(astelor㉿AcerAST)-[/mnt/d/Documents/Quartus/hwextra]
└─$ cat test_dec.txt
13816
46995
37333
43061
16692
14982
11709
8274
8374
5091
44308
23825
7183
15583
31409
40455
52139
4801
28534
57536

┌──(aste
└─$
```

```
┌──(astelor㉿AcerAST)-[/mnt/d/Documents/Quartus/hwextra]
└─$ cat test_hex.txt
0x35f8
0xb793
0x91d5
0xa835
0x4134
0x3a86
0x2dbd
0x2052
0x20b6
0x13e3
0xad14
0x5d11
0x1c0f
0x3cdf
0x7ab1
0x9e07
0xcbab
0x12c1
0x6f76
0xe0c0

┌──(astelor㉿AcerAST)-[/mnt/d/Documents/Quartus/hwextra]
└─$
```

主程式

```
1  module hwextra #(parameter WIDTH = 32)(
2      input clk,
3      input reset,
4      input FIFO_en, // flag that data output
5      input FIFO_w_en, // flag that controls data input
6      input sort_en,
7      input add_en,
8      input [WIDTH-1:0] data,
9      output wire [WIDTH-1:0] FIFO_out,
10     output wire FIFO_empty,
11     output wire [WIDTH-1:0] odd,
12     output wire [WIDTH-1:0] even,
13     output wire [WIDTH-1:0] sum
14 );
15
16 FIFO n1(clk, reset, FIFO_en, FIFO_w_en, data, FIFO_out, FIFO_empty);
17 sort n2(clk, reset, sort_en, FIFO_out, odd, even);
18 adder n3(clk, reset, add_en, FIFO_out, FIFO_empty, FIFO_en, sum);
19
20 endmodule
21
```



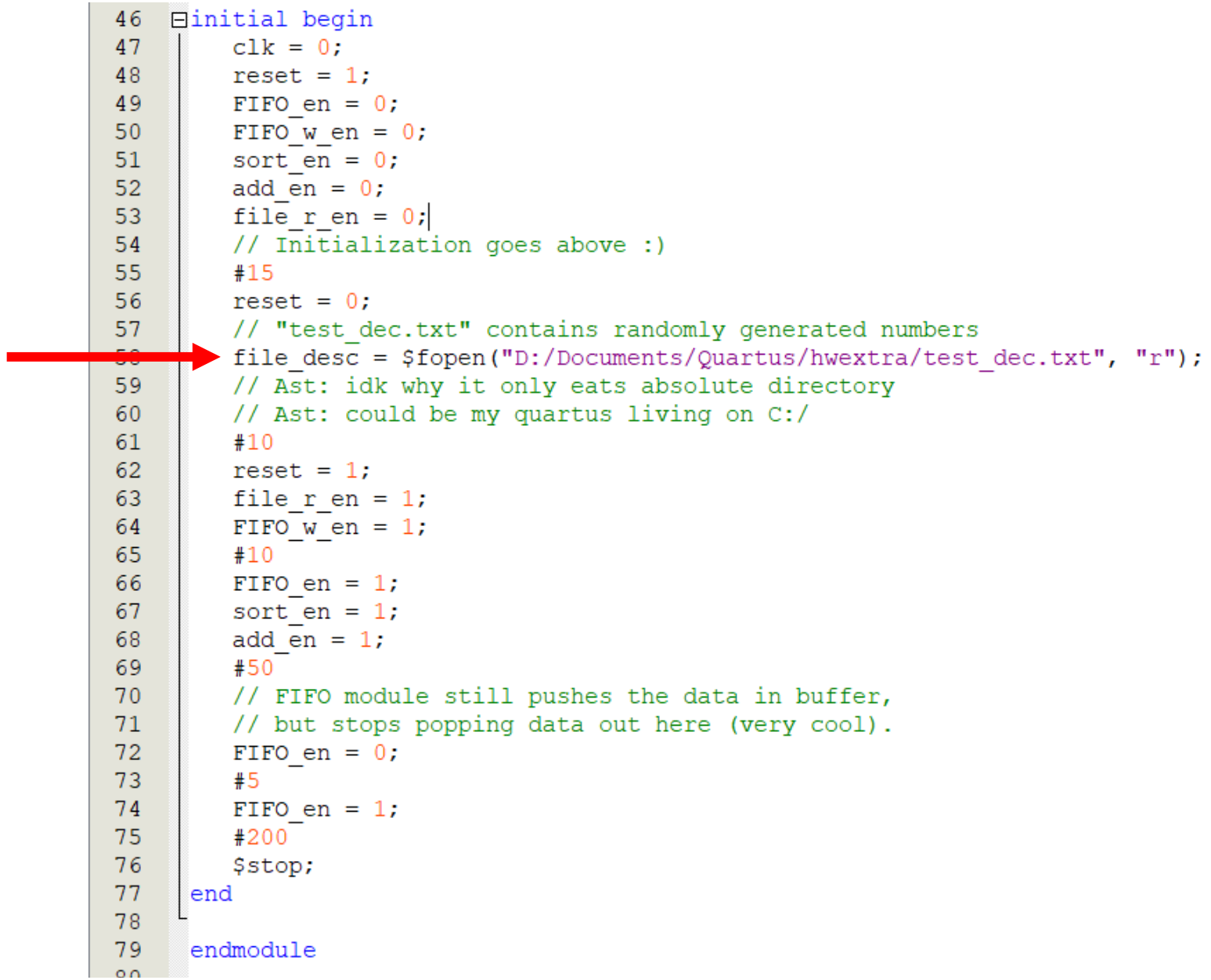
Testbench 1/2

- 系統檔案讀取指令($fscanf 等)不是 synthesizable code，他不能被包括在其他Verilog HDL檔內，只能存在於testbench。

```
1  module hwextra_tb #(parameter WIDTH = 32)();
2
3  // Corresponding ports:
4  reg clk;
5  reg reset;
6  reg FIFO_en;
7  reg FIFO_w_en;
8  reg sort_en;
9  reg add_en;
10 reg [WIDTH-1:0] data;
11 wire [WIDTH-1:0] FIFO_out;
12 wire FIFO_empty;
13 wire [WIDTH-1:0] odd;
14 wire [WIDTH-1:0] even;
15 wire [WIDTH-1:0] sum;
16
17 // Top module import:
18 hwextra p1(clk, reset, FIFO_en, FIFO_w_en, sort_en, add_en, data, FIFO_out, FIFO_empty, odd, even, sum);
19
20 always begin
21    #5 clk = ~clk;
22 end
23
24 // Variables for testbench file reading:
25 integer file_desc; // file descriptor
26 integer file_stat; // value = 1 -> open; value = -1 -> closed;
27 reg file_r_en; // enable text file reading
28
29 always@(posedge clk)begin
30    // read the data in test_dec.txt line by line
31    if(file_desc) begin // check if the file is opened successfully
32       if(file_r_en)begin
33          file_stat <= $fscanf(file_desc, "%d\n", data);
34       end
35       if($feof(file_desc))begin
36          $fclose(file_desc);
37       end
38       if(file_stat==-1)begin
39          FIFO_w_en <= 0;
40          // Stops data input for FIFO module on the next clock,
41          // when the file is closed.
42       end
43    end
44 end
```

Testbench 2/2

- 在第58行，是用來開啟文字檔的程式碼($fopen)，其中的文字檔路徑是用**絕對路徑**。
- 因為我不知道為什麼ModelSim一直讀不到相對路徑，所以使用上需要修改這行程式碼。



```
46  initial begin
47      clk = 0;
48      reset = 1;
49      FIFO_en = 0;
50      FIFO_w_en = 0;
51      sort_en = 0;
52      add_en = 0;
53      file_r_en = 0;
54      // Initialization goes above :)
55      #15
56      reset = 0;
57      // "test_dec.txt" contains randomly generated numbers
58      file_desc = $fopen("D:/Documents/Quartus/hwextra/test_dec.txt", "r");
59      // Ast: idk why it only eats absolute directory
60      // Ast: could be my quartus living on C:/
61      #10
62      reset = 1;
63      file_r_en = 1;
64      FIFO_w_en = 1;
65      #10
66      FIFO_en = 1;
67      sort_en = 1;
68      add_en = 1;
69      #50
70      // FIFO module still pushes the data in buffer,
71      // but stops popping data out here (very cool).
72      FIFO_en = 0;
73      #5
74      FIFO_en = 1;
75      #200
76      $stop;
77  end
78
79  endmodule
```

FIFO

- FIFO_en 是致能訊號，用來控制FIFO模組的輸出。
- 此模組的輸入為負緣觸發，輸出為正緣觸發，如此設計能加速FIFO模組的運作速度。
- 這裡我是用**環狀佇列**完成first-in-first-out的輸入輸出功能，因為環狀佇列能有效運用queue的空間。

```
1  module FIFO #(parameter WIDTH = 32, parameter DEPTH = 10)(
2      input clk,
3      input reset,
4      input enable_out, // flag that allows data popping out of the queue
5      input enable_in, // flag that allows data pushing into the queue
6      input [WIDTH-1:0] data,
7      output reg [WIDTH-1:0] FIFO_out,
8      output reg empty // empty flag to control the adder
9  );
10 reg [WIDTH-1:0] queue [0:DEPTH-1]; //circular queue
11 reg [7:0] head;
12 reg [7:0] tail;
13 reg full;
14
15 // pop out the queue if enable is on until the queue is empty
16 always@(posedge clk or negedge reset)begin
17       if(!reset)begin
18          head <= 0;
19          full <= 0;
20          FIFO_out <= 0;
21       end
22       else begin
23          if (!empty && enable_out)begin
24             FIFO_out <= queue[head];
25             head <= (head + 1) % DEPTH;
26          end
27          else begin
28             FIFO_out <= FIFO_out;
29             head <= head;
30          end
31          // check and update the "full" flag
32          full <= ( (tail + 1) % DEPTH == head);
33       end
34 end
35
```

```
36 // push data into the queue
37 always@(negedge clk or negedge reset)begin
38    if(!reset)begin
39       tail <= 0;
40       empty <= 1;
41       queue[0] <= 0;
42    end
43    else begin
44       if(!full && enable_in)begin
45          queue[tail] <= data;
46          tail <= (tail + 1) % DEPTH;
47       end
48       else begin
49          queue[tail] <= queue[tail];
50          tail <= tail;
51       end
52       // check and update the "empty" flag
53       empty <= (head == tail);
54    end
55 end
56
57 endmodule
58
```

Adder

- 在此導入FIFO_empty，偵測輸出是否已結束，避免FIFO模組已經結束輸出，但adder模組仍重複將FIFO模組最後一個輸出數值相加。
- 在此導入FIFO_en，是為了避免FIFO模組無輸出但adder仍相加數值的問題。

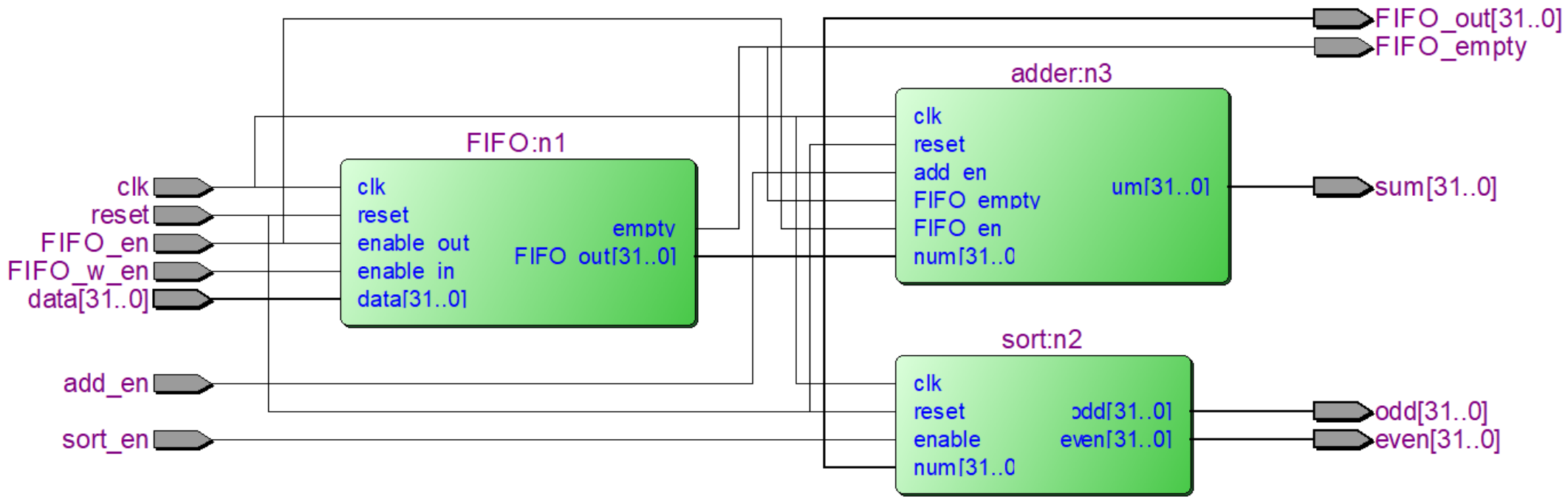
```
1  module adder #(parameter WIDTH = 32)(
2      input clk,
3      input reset,
4      input add_en,
5      input [WIDTH-1:0] num,
6      input FIFO_empty,
7      input FIFO_en, // adder is tied with FIFO module
8      output reg [WIDTH-1:0] sum
9  );
10
11 always@(posedge clk or negedge reset)begin
12     if(!reset)begin
13         sum <= 0;
14     end
15     else begin
16         if(add_en && !FIFO_empty && FIFO_en)begin
17             sum <= sum + num;
18         end
19         else begin
20             sum <= sum;
21         end
22     end
23 end
24
25 endmodule
26
```

Sort

- 將FIFO模組的輸出分為奇數或偶數。

```
1  module sort #(parameter WIDTH = 32)(
2      input clk,
3      input reset,
4      input enable, // sort enable
5      input [WIDTH-1:0] num,
6      output reg[WIDTH-1:0] odd,
7      output reg[WIDTH-1:0] even
8  );
9
10 always@(posedge clk or negedge reset)begin
11     if(!reset)begin
12         odd <= 0;
13         even <= 0;
14     end
15     else if(enable)begin
16         if(num & 1)begin
17             odd <= num;
18         end
19         else begin
20             even <= num;
21         end
22     end
23     else begin
24         odd <= odd;
25         even <= even;
26     end
27 end
28
29 endmodule
30
```

RTL 圖示


clk
reset
FIFO_en
FIFO_w_en
data[31..0]
add_en
sort_en
FIFO:n1
clk
reset
enable_out
enable_in
data[31..0]
empty
FIFO_out[31..0]
adder:n3
clk
reset
add_en
FIFO_empty
FIFO_en
num[31..0
um[31..0]
sort:n2
clk
reset
enable
num[31..0
odd[31..0]
even[31..0]
FIFO_out[31..0]
FIFO_empty
sum[31..0]
odd[31..0]
even[31..0]

波形模擬圖

```
產生的20個十進位數字：
[ 13816,  46995,  37333,  43061,  16692]
[ 14982,  11709,   8274,   8374,   5091]
[ 44308,  23825,   7183,  15583,  31409]
[ 40455,  52139,   4801,  28534,  57536]

轉換為16進位制的數字：
[0x35f8, 0xb793, 0x91d5, 0xa835, 0x4134]
[0x3a86, 0x2dbd, 0x2052, 0x20b6, 0x13e3]
[0xad14, 0x5d11, 0x1c0f, 0x3cdf, 0x7ab1]
[0x9e07, 0xcbab, 0x12c1, 0x6f76, 0xe0c0]

累加總和（十進位）：512100
累加總和（16進位）：0x7d064
```