Computer-Aided Design for VLSI Design

Homework1（Student ID: 41147046S｜Name: 楊子萱）

1. Provide a simple explanation of your code.

rng_cell.vhd

```
 1  library IEEE;
 2  use IEEE.STD_LOGIC_1164.ALL;
 3
 4  entity rng_cell is
 5      Port (
 6          in_a, in_b, in_c, in_d : in STD_LOGIC;
 7          out_val : out STD_LOGIC
 8      );
 9  end rng_cell;
10
11  architecture Behavioral of rng_cell is
12  begin
13      out_val <= in_a xor in_b xor in_c xor in_d;
14  end Behavioral;
15
```

建立一個單一隨機單元（cell）。

接收來自 4 個相鄰位置的輸入位元（in_a ~ in_d）

使用 XOR 組合邏輯 計算輸出 out_val

rng_array.vhd

```
 1  library IEEE;
 2  use IEEE.STD_LOGIC_1164.ALL;
 3  use IEEE.NUMERIC_STD.ALL;
 4
 5  entity rng_array is
 6      Port (
 7          clk : in STD_LOGIC;
 8          rst : in STD_LOGIC;
 9          seed : in STD_LOGIC_VECTOR(63 downto 0);
10          enable : in STD_LOGIC;
11          rng_out : out STD_LOGIC_VECTOR(63 downto 0)
12      );
13  end rng_array;
```

匯入標準 VHDL 資料類型庫

定義元件 rng_array，這是整個隨機數產生器

seed 是初始輸入（64-bit）

clk, rst, enable 控制更新時機

rng_out 是輸出的 64-bit 隨機數

```
15  architecture Behavioral of rng_array is
16
17      component rng_cell
18          Port (
19              in_a, in_b, in_c, in_d : in STD_LOGIC;
20              out_val : out STD_LOGIC
21          );
22      end component;
23
24      signal cell_out : STD_LOGIC_VECTOR(63 downto 0);
25      signal next_state : STD_LOGIC_VECTOR(63 downto 0);
```

宣告使用 rng_cell 組件

cell_out：目前的狀態值（64-bit）

next_state：下一輪會變成的狀態值

```
29      gen_cells: for i in 0 to 63 generate
30          constant idx_a : integer := (i - 2 + 64) mod 64; -- -2
31          constant idx_b : integer := (i + 1) mod 64;      -- 1
32          constant idx_c : integer := i;                   -- 0
33          constant idx_d : integer := (i + 2) mod 64;      -- 2
34      begin
35          cell_inst : rng_cell
36              port map (
37                  in_a => cell_out(idx_a),
38                  in_b => cell_out(idx_b),
39                  in_c => cell_out(idx_c),
40                  in_d => cell_out(idx_d),
41                  out_val => next_state(i)
42              );
43      end generate;
```

用 **generate-for 產生 64 個 cell**

對每個 cell 指定 4 個輸入的連線來源

使用 mod 64 確保在 0~63 間循環（環狀結構）

```
45      process(clk, rst)
46      begin
47          if rst = '1' then
48              cell_out <= seed;
49          elsif rising_edge(clk) then
50              if enable = '1' then
51                  cell_out <= next_state;
52              end if;
53          end if;
54      end process;
55
56      rng_out <= cell_out;
57
58  end Behavioral;
```

若 rst 為 '1'，就把整個狀態重設成 seed

每當 clock 上升沿時，若 enable = '1'，就把 next_state 複製到 cell_out

cell_out 是目前狀態，用於下一輪 cell 的輸入

rng_tb.vhd

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_TEXTIO.ALL;
4   use STD.TEXTIO.ALL;
5   use IEEE.NUMERIC_STD.ALL;
6
7   entity rng_tb is
8   end rng_tb;
```

匯入 VHDL 標準邏輯與文字輸出函式庫（TEXTIO 和

STD_LOGIC_TEXTIO 用來寫 .txt）

rng_tb 是測試模組（testbench），沒有對外的 Port

```
10  architecture behavior of rng_tb is
11
12      component rng_array
13          Port (
14              clk : in STD_LOGIC;
15              rst : in STD_LOGIC;
16              seed : in STD_LOGIC_VECTOR(63 downto 0);
17              enable : in STD_LOGIC;
18              rng_out : out STD_LOGIC_VECTOR(63 downto 0)
19          );
20      end component;
```

宣告被測試的模組 rng_array

```
22      signal clk : STD_LOGIC := '0';
23      signal rst : STD_LOGIC := '1';
24      signal seed : STD_LOGIC_VECTOR(63 downto 0) := x"DEADBEEFCAFEBABE";
25      signal enable : STD_LOGIC := '0';
26      signal rng_out : STD_LOGIC_VECTOR(63 downto 0);
27
28      constant CLK_PERIOD : time := 10 ns;
```

宣告內部用的訊號

seed 是初始輸入資料

clk、rst、enable 控制模擬流程

rng_out 接收主模組輸出

CLK_PERIOD 用來定義 clock 的週期（這裡是 10ns）

```
32      uut: rng_array port map (
33          clk => clk,
34          rst => rst,
35          seed => seed,
36          enable => enable,
37          rng_out => rng_out
38      );
```

把 rng_array 實體化為 uut

把 testbench 裡的訊號接到元件 port 上

```
40      clk_process :process
41      begin
42          clk <= '0';
43          wait for CLK_PERIOD/2;
44          clk <= '1';
45          wait for CLK_PERIOD/2;
46      end process;
```

模擬一個 10ns 週期的 clock（5ns low + 5ns high）

```
48      stim_proc: process
49          file out_file : text open write_mode is "output_data2.txt";
50          variable linebuf : line;
```

開啟一個文字檔（寫入模式），準備寫入每一個 clock cycle 的輸出結果

使用 linebuf 這個 buffer 暫存每一行的字串內容

```
52          wait for 20 ns;
53          rst <= '0';
54          enable <= '1';
```

等待 20ns（讓系統 reset 生效）

接著關閉 reset，啟動 enable，使主模組可以開始演算

```
56          for i in 0 to 19 loop
57              wait until rising_edge(clk);
58              write(linebuf, string'("Cycle "));
59              write(linebuf, i);
60              write(linebuf, string'(": "));
61              write(linebuf, rng_out);
62              writeline(out_file, linebuf);
63          end loop;
```

模擬 20 個 clock cycle，每次 rising_edge 時：

把目前的 rng_out 輸出寫入到 linebuf

接著把這行寫到 output_data2.txt

2. Waveform diagram here (Simulation Results)

(a) 的結果

```
Cycle 0: 1101111010101101101111101110111111001010111111101011101010111110
Cycle 1: 1011101100001011011100111111110001001000110000110011110000110011
Cycle 2: 1111111001011010111100010000010111101111000101000001010101000001
Cycle 3: 1000001001010100110101111100110110111101011100110011000000110010
Cycle 4: 0110011101000010001001100100001101110110011100000000010001000010
Cycle 5: 0000001110110111111100000111010011111100001101000000111111110111
Cycle 6: 0100010111101110000101001011101001000101010010110001010000011111
Cycle 7: 1011110110111110101100101011110101111100001010100111001100101001
Cycle 8: 0111011101110011001000100011011001100101011000010011000000100100
Cycle 9: 1011111111110000001111111100110000000100010010111000010001110111
Cycle 10: 1111000000010100010100000100000100001111111010111010111110111110
Cycle 11: 1101010000110011110011001111001111010100001100111100110011110011
Cycle 12: 0110001101000001010000000101000101100011010000010100000001010001
Cycle 13: 0100110010110011001100001100111101001100101100110011000011001111
Cycle 14: 1010000010100000000001010000010110100000101000000000010100000101
Cycle 15: 0101100110011000000011001100110101011001100110000000110011001101
Cycle 16: 0001000000000010000100000000001000010000000000010000100000000010
Cycle 17: 1011110000000111101111000000011110111100000001111011110000000111
Cycle 18: 1111010100001010111101010000101011110101000010101111010100001010
Cycle 19: 1101100011011000110110001101100011011000110110001101100011011000
```

(b) 的結果

```
Cycle 0: 1101111010101101101111101110111111001010111111101011101010111110
Cycle 1: 0111110011100110111101011001110011110110110001001010001110110101
Cycle 2: 0110111000110111111001000100011001101011000001010101010001010111
Cycle 3: 1011101001111110011111100110110101101000010100111111101000110100
Cycle 4: 0010000000100111001001110111010100100111001000010001000101110001
Cycle 5: 0111100010100001111000001100011001100001111111011001100001010001
Cycle 6: 0111100001011110111010111000110101001110100010100101001100101000
Cycle 7: 1011101100011101100010101001110110000000010001001010001011010111
Cycle 8: 0110010000100010000001110000001000100001011001010101000001010111
Cycle 9: 1001111110110011100110010100101100111100000110111111011010011010
Cycle 10: 1000100101001000110101101001000010011101011111110101010000110001
Cycle 11: 0000101010011101110100100011111010000001011000100111101111100011
Cycle 12: 0110011100000011000001110101010001100100000000100011011101100110
Cycle 13: 0101000101001110010110000111101011011111000010110111111001001010 1
Cycle 14: 1110100010000011100000111011001001111011011000011110111110101111
Cycle 15: 0100010011101100101011000100111000110101100011101101110100101100
Cycle 16: 0110010000010011011000000100001001110100000100010111000001000000
Cycle 17: 1101111101010011100010010111101000000011010110000101010101110001
Cycle 18: 0011101001100000100010100111000110001111010000110011111101010010
Cycle 19: 0100000001001010110001000001001100010110001111101001001001100111
```

3. Reflections and discussions

這次作業讓我實際練習了如何使用 VHDL 設計並模擬 1D cellular automata 結構的隨機數產生器（RNG）。我學會了如何用 generate 搭配 mod 運算來實現環狀連接，並利用 XOR 組合邏輯設計基本的 rng_cell。透過

ModelSim 測試，我也理解了 testbench 的撰寫方式、clock 與 reset 的控制邏輯，以及如何使用 TextIO 將模擬結果輸出為 .txt 檔。整體過程加深了我對硬體行為模擬與時序控制的理解，也訓練了系統化 debug 的能力。