Computer-Aided Design for VLSI Design

Homework4（Student ID: 61047046S｜Name: 林宣佑）

1. Provide a simple explanation of your code.

這個專案的目標是實現一個隨機數生成器，使用 CA.vhd、RNG.vhd 和 test_bench_3.vhd 三個元件共同合作。

CA.vhd：這個元件代表一個有限自動機，根據輸入的 adr（地址）信號選擇相應的輸出 Y。

RNG.vhd：這個元件利用 CA.vhd 元件組成一個反饋移位寄存器，通過將前一個輸出作為輸入並進行適當的位元操作，生成隨機數。

test_bench_3.vhd：這個元件是測試環境，用於驅動 RNG 元件並模擬數據的輸入和輸出。

CA：

在 entity CA 中定義了它的實體。它具有五個 port：

- adr (位址)：輸入訊號，用於選擇特定的輸入資料。
- clk (時脈)：輸入訊號，用於驅動整個系統的時脈。
- sel (選擇)：輸入訊號，用於控制是否進行初始化。
- init (初始值)：輸入訊號，用於設定初始化值。
- Y (輸出)：輸出訊號，代表電路的輸出結果。

在 architecture behavioral of CA 中，先是定義了訊號 X，根據 adr 的值，選擇輸出並存入 X 中。利用 with 的語法，可以根據不同的 adr 利用查表的方式找出對應的輸出。以我這次的程式碼為例，adr 為 0000 時，便會將 X 的值設為 0。

接下來我們定義了一個 process，並偵測 clk 訊號。當 rising_edge 時，若 sel 為 1 即會將 X 的值指派給輸出信號 Y；反之若 sel 為 0 則會將初始值 init 指派給 Y。

RNG：

在 entity RNG 中定義了它的實體。它具有四個 port：

- seed (種子)：輸入訊號，64 位元的種子輸入，用於初始狀態。
- clk (時脈)：輸入訊號，用於驅動整個系統的時脈。
- sel (選擇)：輸入訊號，選擇輸入，用於控制亂數生成。
- rng_out (初始值)：輸出訊號，代表隨機數生成器產生的隨機數值。

在 architecture structure of RNG 中，主要是將既有電路組合起來，故 RNG 為 structure model。首先宣告了一個型別為 CA_input，用於儲存 64 個 CA 的陣列，並隨之將先前介紹的 CA 作為 component 引入。

接著定義兩個 signal：作為儲存陣列的 my_CA_input 以及接收所有 CA 輸出的 my_CA_output。

在 processs 中我們偵測 my_CA_output 若有變化時，使用 for 迴圈根我們的規則產生 i、j、k、l 四個變數並將之組合成一個 std_logic_vector 寫入 my_CA_input 的每個 CA 中。

結束 proccess 的定義後透過 CA_generate 區塊的程式碼將所有 CA 將其對應的訊號組合，CA 的 adr 由 my_CA_input(n)提供，clk 和 sel 由外部輸入的 clk 和 sel 提供，init 由 seed(n)提供，最後所有 CA 的 Y 連接到 my_CA_output(n)並組合成 my_CA_output。

最後將 my_CA_output 寫入 rng_out 並作為 RNG.vhd 的輸出。

test_bench_3：

Test_bench_3 的設計是為了用於驗證 RNG 的元件功能，在 entity Test_bench_3 中定義了它的實體。

在 architecture arch_test of CA 中。我們定義了四個訊號：

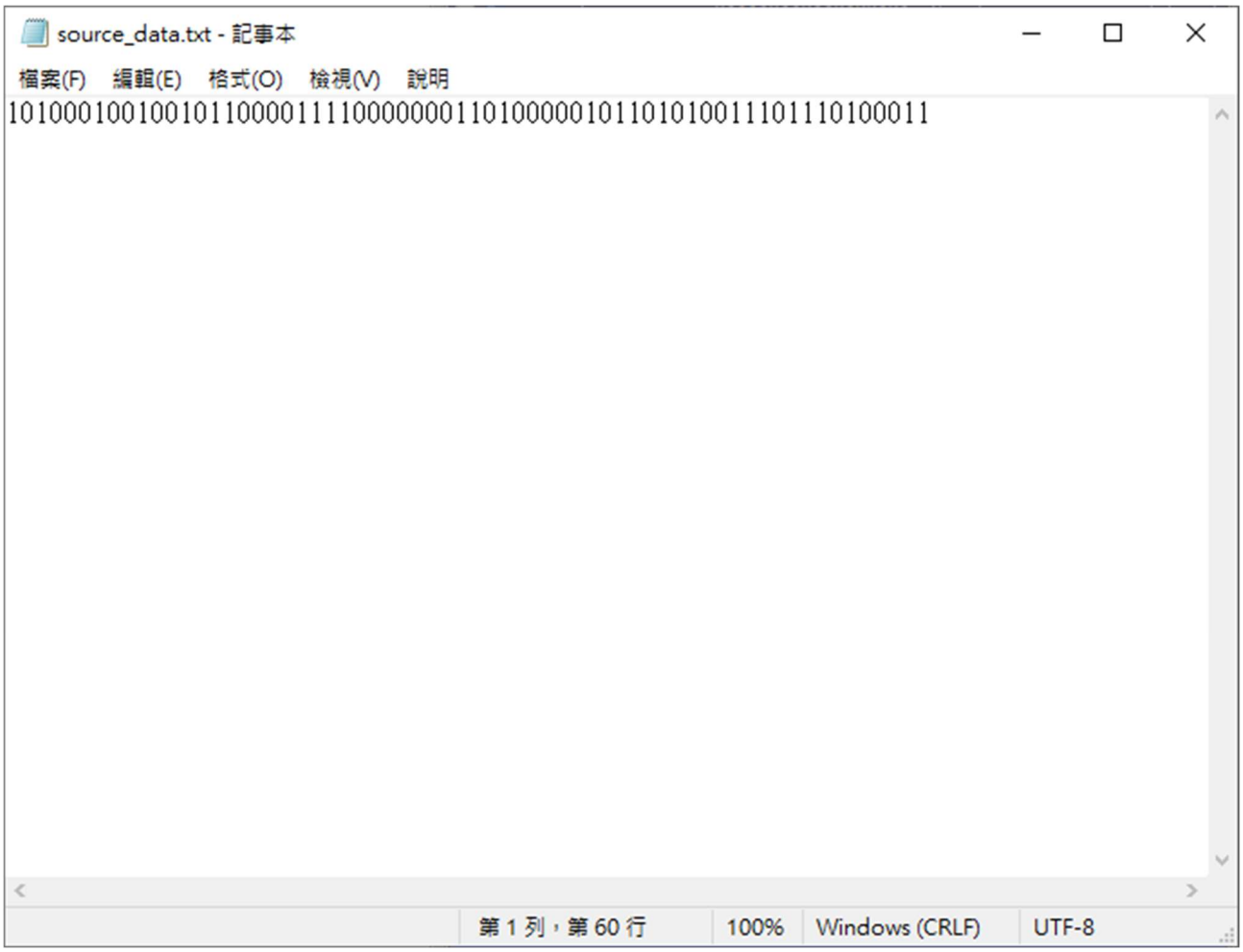
- seed(種子)：64-bit 的 random seed，從 source_data.txt 提取。
- clk (時脈)：用於驅動整個系統的時脈，100ns 為一個週期。
- sel (選擇)：起先設定 0 以初始化，在 100ns 時改變為 1。
- rng_out (輸出)：經測試由 RNG 元件輸出的 64-bit 隨機數。

接著將以上訊號連接到 RNG 元件相對應的端口上，以模擬對 RNG 元件的輸入以及記錄他的輸出，測試過程必須先打開 source_data.txt，透過

readline 函式從其讀取一行數字，並使用 read 將其轉換為 64-bit 的型態指派給 temp，最後將 temp 指派給 seed。

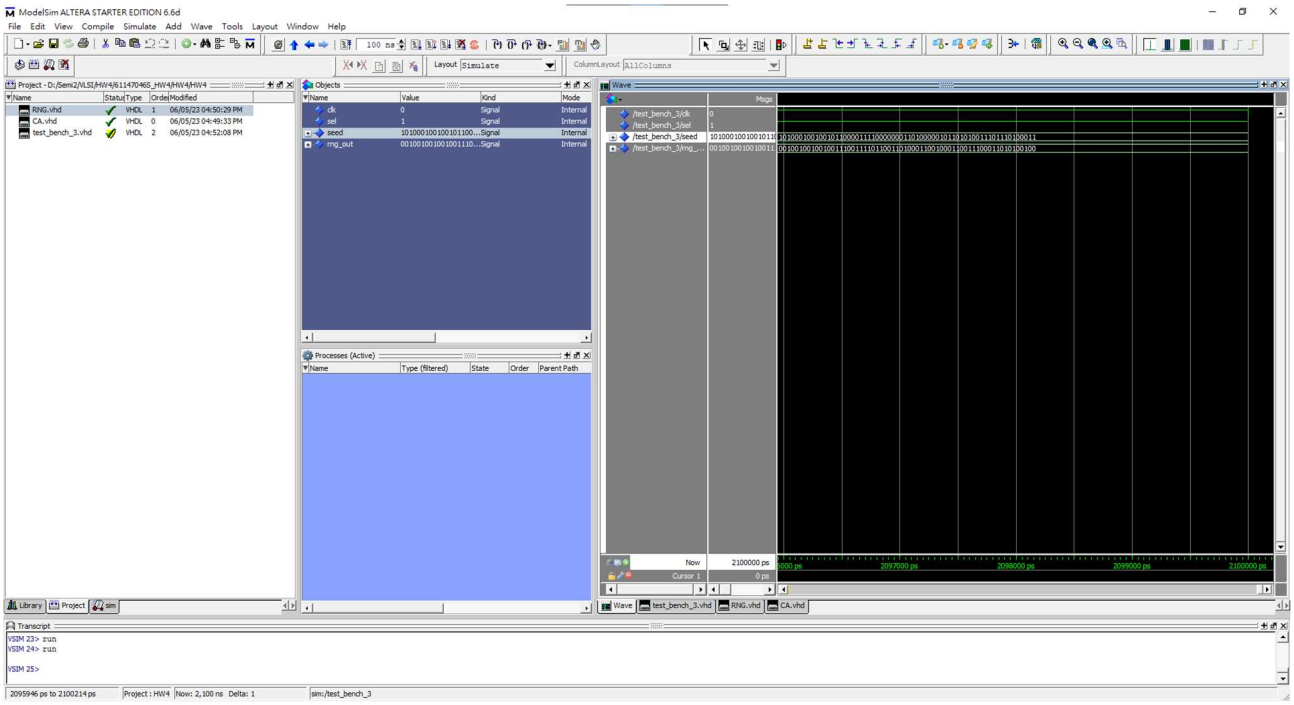
待 RNG 運作後輸出隨機數，我們先檢查 rng_out 是否有發生變化且不全等於 U。若滿足條件則透過 write 函式將 rng_out 寫入 out_data，再透過 writeline 函式將 out_data 輸入至 output_data.txt 中。

2. Waveform diagram here (Simulation Results)



圖表 1 Source_data

圖表 2 Output_data



圖表 3 Modelsim 截圖

3. Reflections and discussions

在這次的專案中，我們建立了一個包含三個主要元件的組合。首先，我們有 CA.vhd，它代表了元件自動機（Cellular Automaton）的行為，根據給定的位址（adr）產生對應的輸出信號（Y）。接下來，我們有 RNG.vhd，它使用多個 CA 元件組成的環形結構，從而生成隨機的輸出信號（rng_out）。最後，我們透過 test_bench_3.vhd 測試這些元件的功能。它使用一個時脈信號（clk）來驅動模擬的時序行為，並在一開始將 sel 訊號設定為 0 將所有 CA 元件進行初始化。同時，從一個文字檔中讀取我們設定的 random seed，將其作為輸入提供給 RNG 元件，並將 rng_out 的結果寫入另一個文字檔作為模擬結果。

這次作業使用到我們這學期所學到所有的 VHDL 技能，從設計單個元件到可以將許多元件組成為得以實現隨機數產生器的電路，同時可以透過 test bench 來驗證元件的正確性以及隨機數生成的結果。

4. Optional

在加分題項目中必須透過 HW4 的 RTL 元件，將之產生的隨機結果經過 module 後轉換為 0~20 的數字，並取連續三個做為樂透號碼。

我撰寫了一隻 Optional.vhd 的主程式，在 entity Optional 中定義了它的實體。它具有五個 port：

- clk (時脈)：輸入訊號，用於驅動整個系統的時脈。
- en (開啟)：輸入訊號，用於選擇特定的輸入資料。

- sel (選擇)：輸入訊號，用於控制是否進行初始化。
- seed (種子)：輸入訊號，64 位元的種子輸入，作為 random seed。
- result (輸出)：輸出訊號，代表輸出的樂透號碼。

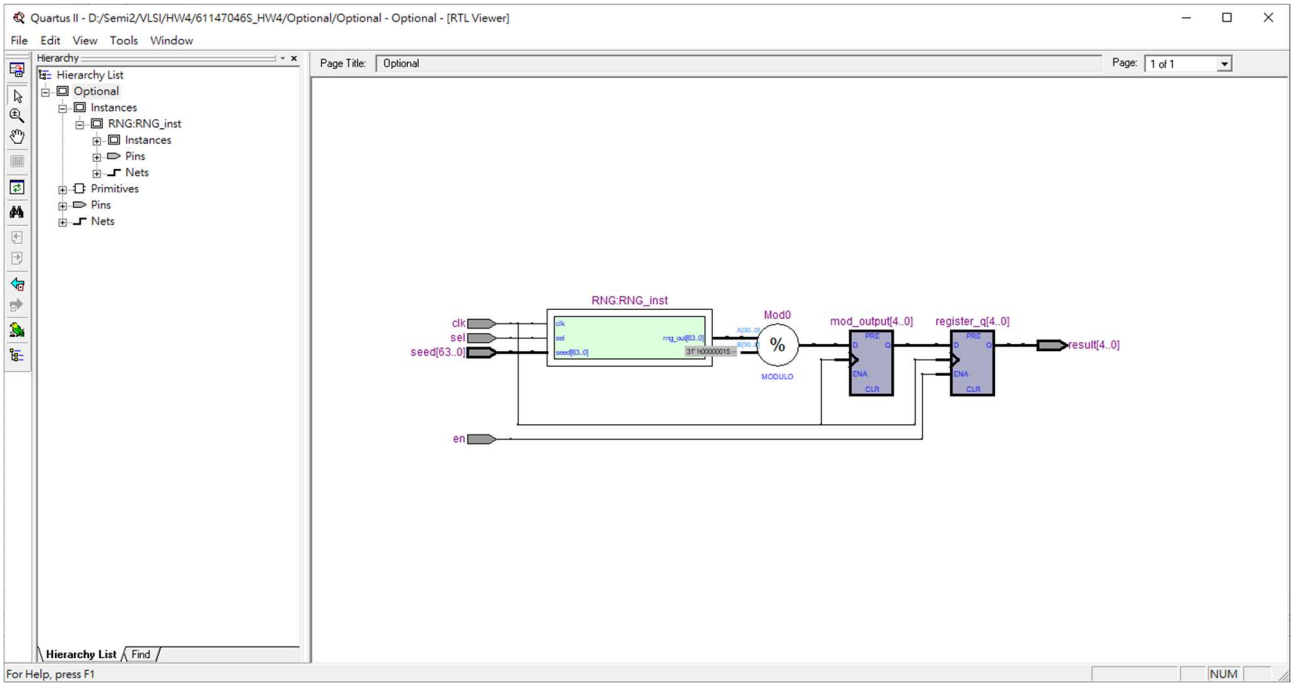
在 architecture behavior of Optional 中定義了它的行為。首先將 RNG 作為 component 引入，再來定義了三個訊號：rng_output 紀錄 RNG 的輸出、mod_output 紀錄 rng_output 經 module 計算後的結果、regitster_q 作為 D 型正反器紀錄最終的輸出結果。

在 process 中我們會定義一個名為 dec_value 的 integer，以作為計算 rng_output 的中繼型態，當 rising_edge 觸發時將 rng_output 轉換為無符號數再將之轉換為 integer 記錄至 dec_value 中，接下來將 dec_value 作 module 運算並將之轉換回 unsigned 型態最後再轉回 std_logic_vector 型態寫入 mod_output 中。

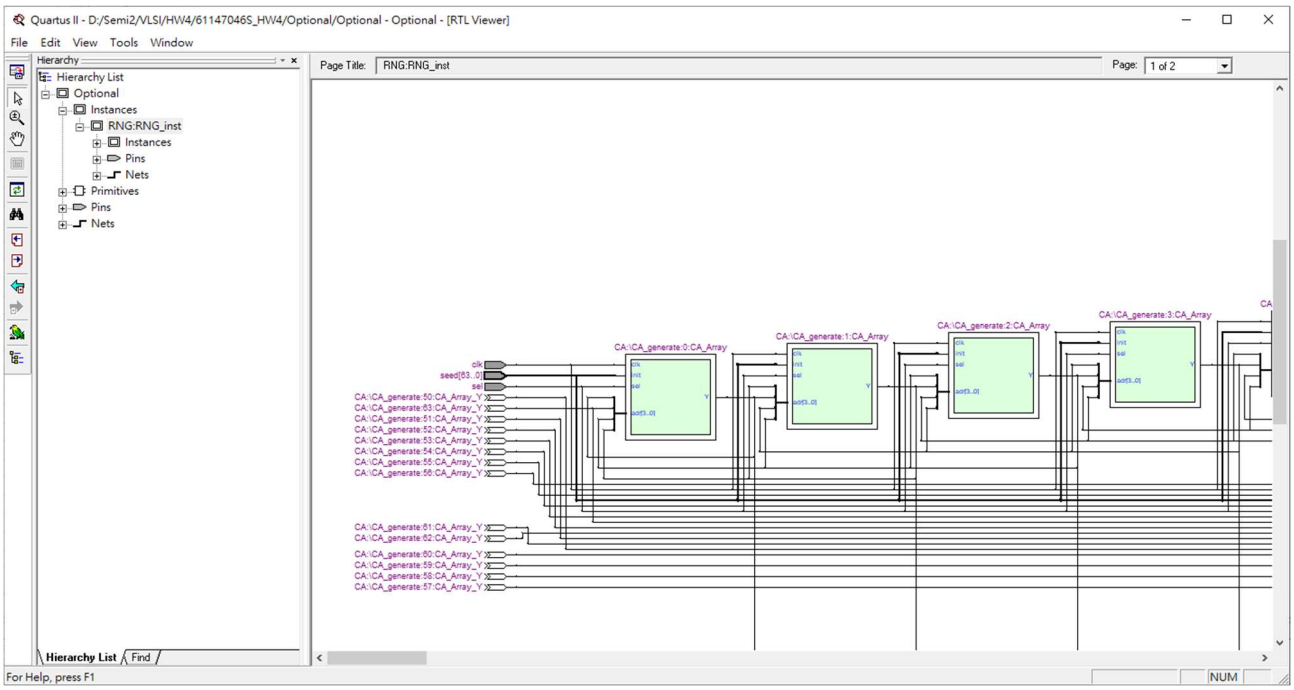
下個階段的 process 則是在 rising_edge 觸發時確認 en 訊號是否允許將運算結果寫入 D 型正反器，若 en 為一就將結果寫入 register_q 中。

最後將 D 型正反器的內容輸出。

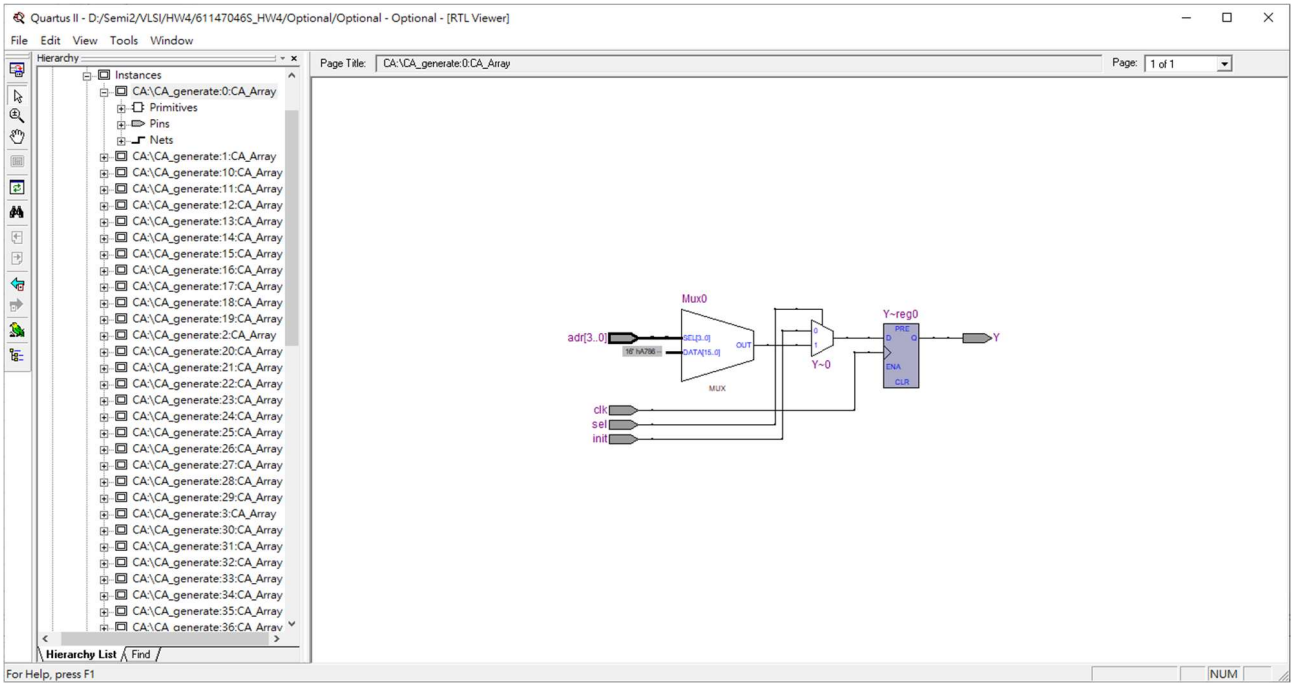
附圖為產生出的 RTL 電路圖：



圖表 4 Optional

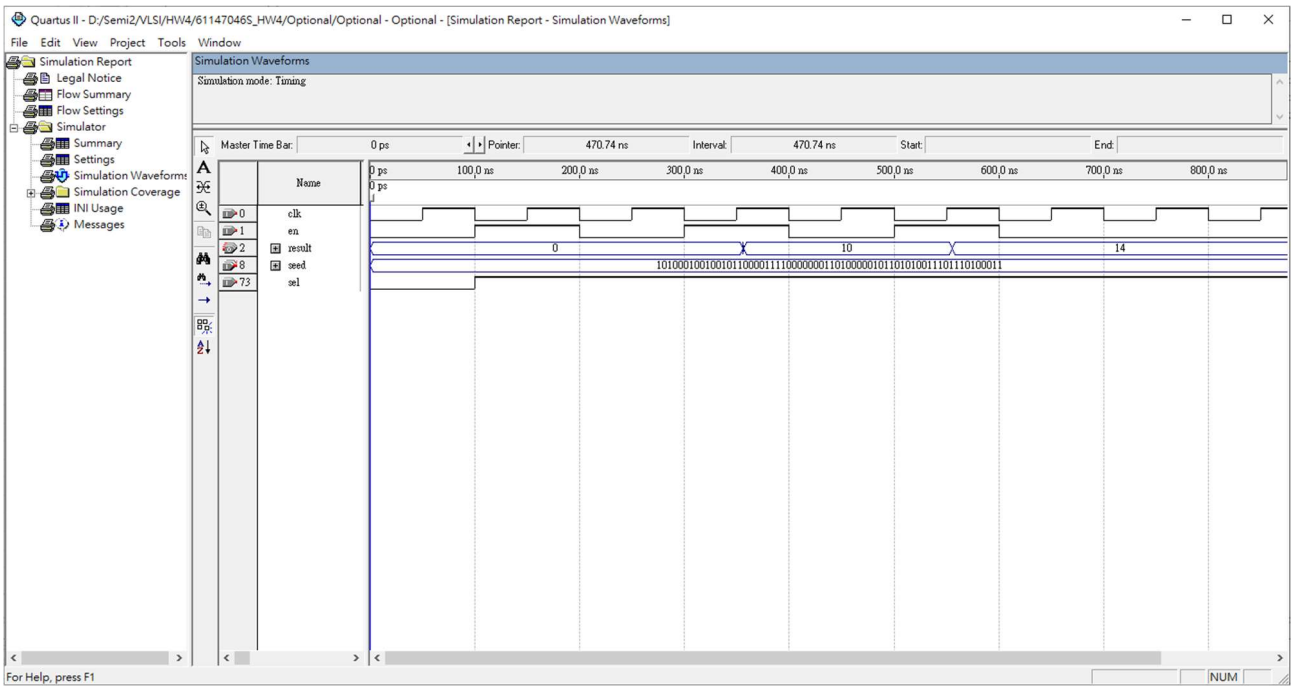


圖表 5 RNG



圖表 6 CA

最後是 Optional 產生的波形圖：



圖表 7 波形圖

可以觀察到當 en='1'時，result 即會有不同的隨機數輸出，且經過 3 個 en 的 rising_edge 後，因為 en='0'，所以 result 就不會再有變化，此次實驗輸出的樂透號碼為：0、10、14。