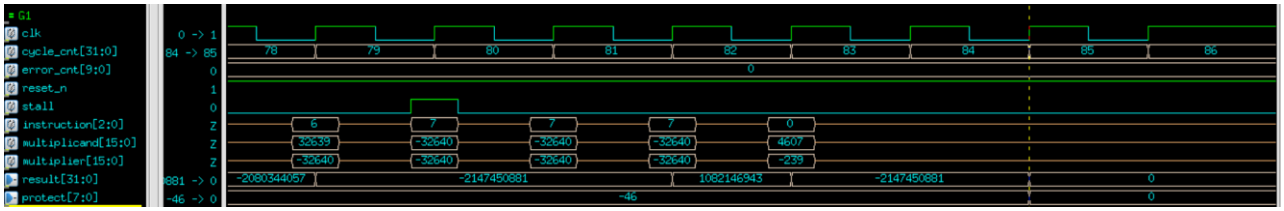
實驗結果圖：（波形圖及模擬完成截圖）



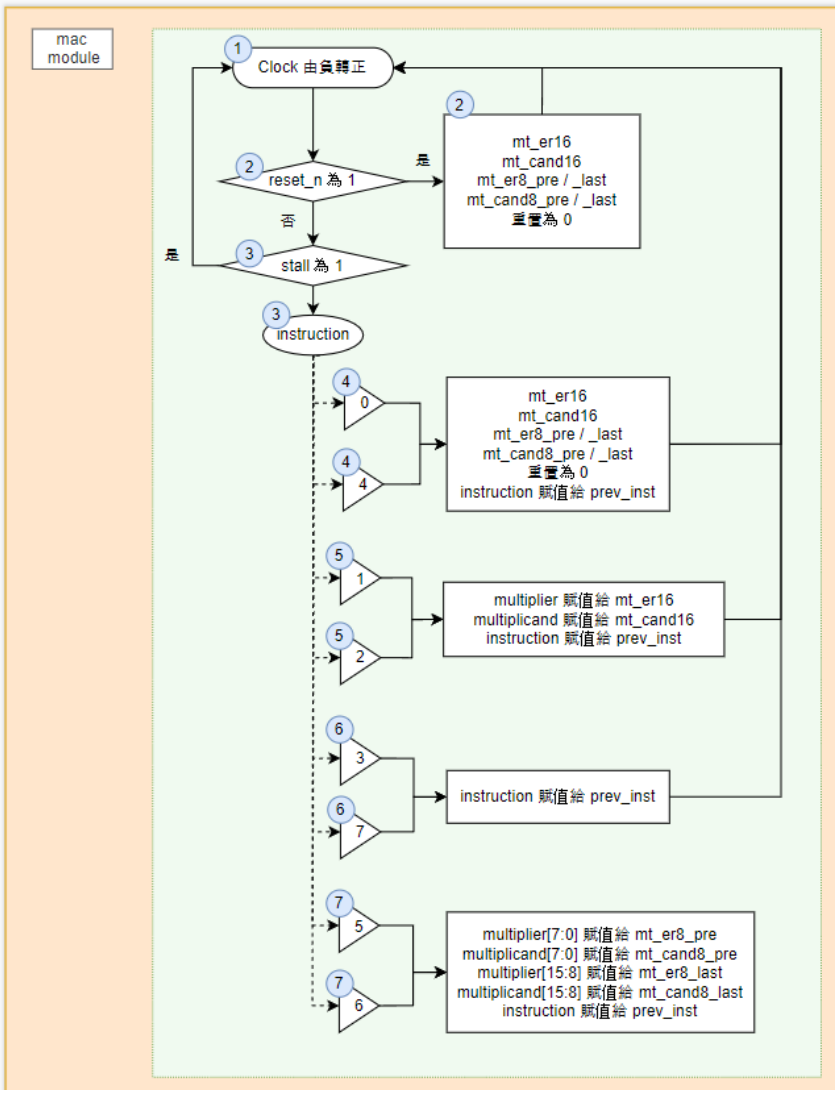
（波形圖）

```
*Verdi* FSDB WARNING: The FSDB file already exists. Overwriting the FSDB file may crash the programs that are using this file.
*Verdi* : Create FSDB file 'mac.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
start test
------------------------------
a1. test MUL_16, cycle = 00000001
------------------------------
a2. test MAC_16, overflow, cycle = 00000005
------------------------------
a3. test stall, cycle = 00000011
------------------------------
a4. test SAT_16, cycle = 00000013
------------------------------
a5. test MAC_16, cycle = 00000015
------------------------------
a6. stall insertion, cycle = 00000021
a7. test stall, cycle = 00000026
a8. test SAT_16, cycle = 00000027
------------------------------
b1. test MUL_8, cycle = 0000002a
------------------------------
b2. test MAC_8, overflow, cycle = 0000002e
------------------------------
b3. test stall, cycle = 0000003a
------------------------------
b4. test SAT_8, cycle = 0000003c
------------------------------
b5. test MAC_8, cycle = 0000003e
------------------------------
b6. stall insertion, cycle = 0000004a
b7. test stall, cycle = 0000004f
b8. test SAT_8, cycle = 00000051
================================================
Congratulation!, Your design PASSed all the test patterns
================================================
Simulation complete via $finish(1) at time 1744 NS + 0
../sim/testfixture.v:172      $finish;
ncsim> exit
➜  HW3_P110775004_online git:(main) ✗
```

（make rtl）

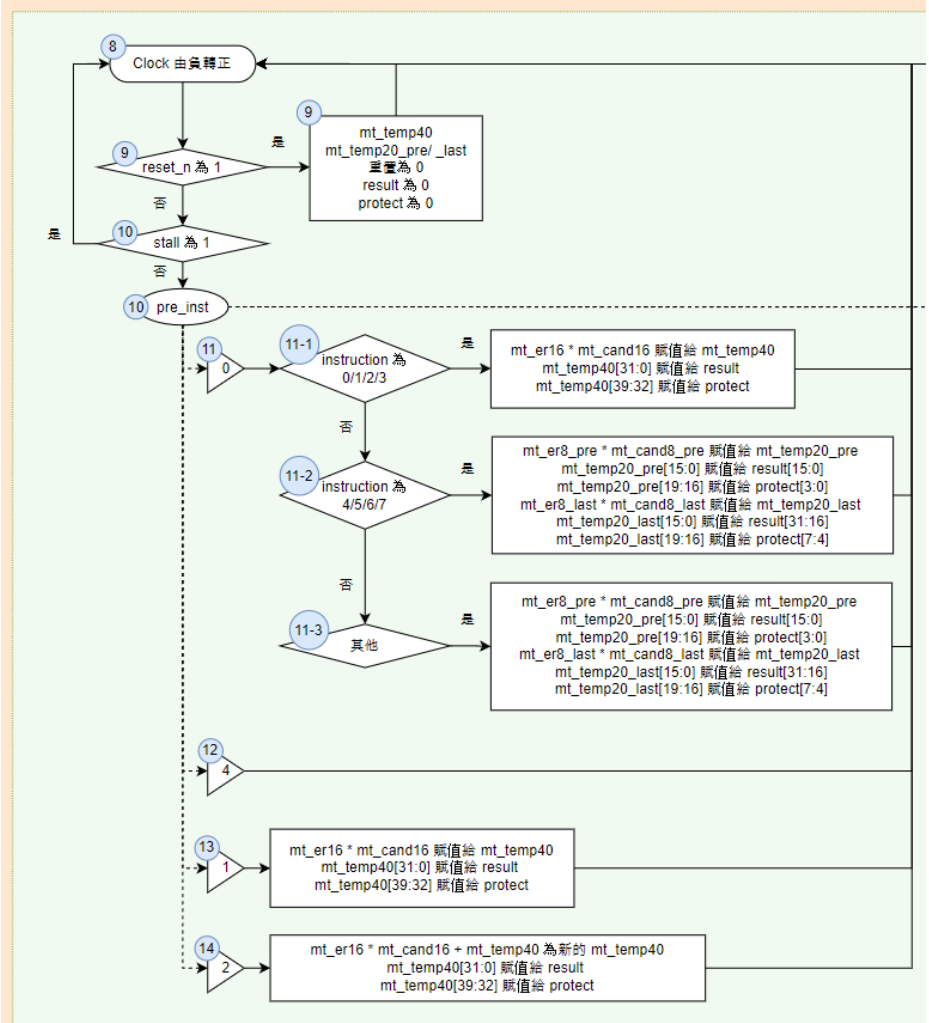
程式運作流程：

（簡單說明波形變化的意義）



【第一個循序網路】

1. 每當 clock 由負轉正時觸發
2. 判定是否 reset，是則重置該網路所使用的 reg，否則繼續往下
3. 判定是否 stall，是則不動作，等待下個 clock，否則繼續往下判斷 instruction 的值
4. 若為 0 或 4，重置該網路所使用的 reg，並將 instruction 賦值給 prev_inst
5. 若為 1 或 2，兩個 input 賦值給 mt_er16 & mt_cand16，並將 instruction 賦值給 prev_inst
6. 若為 3 或 7，將 instruction 賦值給 prev_inst
7. 若為 5 或 6，兩個 input 對拆賦值給 mt_er8_pre / _last & mt_cand8_pre / _last，並將 instruction 賦值給 prev_inst



【第二個循序網路-1】

8. 每當 clock 由負轉正時觸發

9. 判定是否 reset，是則重置該網路所使用的 reg，且兩個 output（protect & result）為 0；否則繼續往下

10. 判定是否 stall，是則不動作，等待下個 clock；否則繼續往下判斷 pre_inst 的值

11. 若為 0，接著判斷 instruction 的值:

    11-1. 若為 0/1/2/3，mt_er16 & mt_cand16 相乘賦值給 mt_temp40，其後 32 位數賦值給 result，前 8 位數賦值給 protect

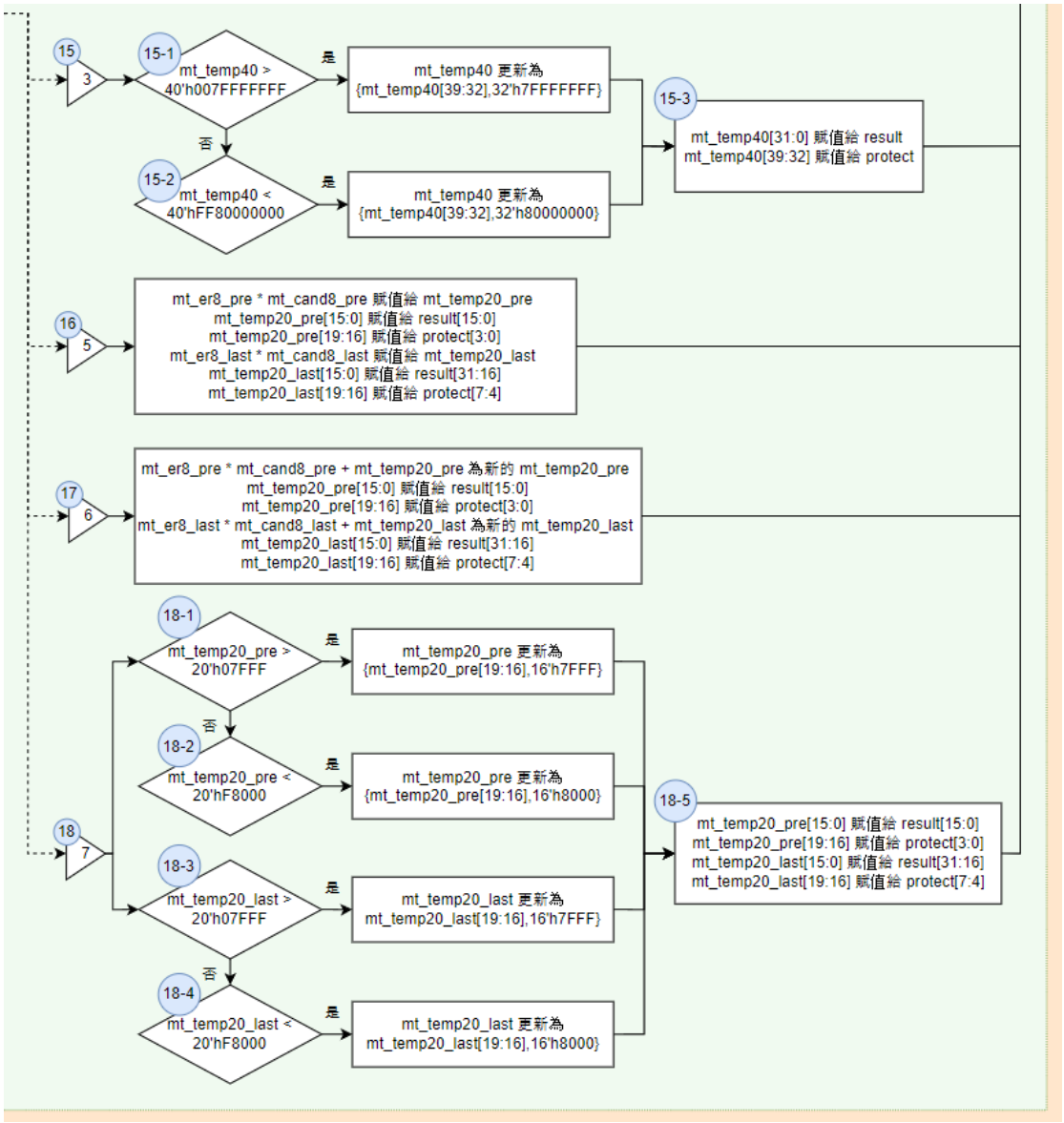
    11-2. 若為 4/5/6/7，mt_er8_pre & mt_cand8_pre 相乘賦值給 mt_temp20_pre，其後 16 位數賦值給 result 後 16 位數，前 4 位數賦值給 protect 後 4 位數

    mt_er8_last & mt_cand8_last 相乘賦值給 mt_temp20_last，其後 16 位數賦值給 result 前 16 位數，前 4 位數賦值給 protect 前 4 位數

    11-3. 若為其他情況，比照 instruction 為 4/5/6/7 的作法

12. 若為 4，不動作

13. 若為 1，mt_er16 & mt_cand16 相乘賦值給 mt_temp40，其後 32 位數賦值給 result，前 8 位數賦值給 protect
14. 若為 2，mt_er16 & mt_cand16 相乘加上原本的 mt_temp40 更新 mt_temp40，其後 32 位數賦值給 result，前 8 位數賦值給 protect



【第二個循序網路-2】

15. 若為 3，接著判斷 mt_temp40 的值:
    15-1. 若為大於 40'h007FFFFFFF，mt_temp40 後 32 位數更新為 32'h7FFFFFFF（32 位有號數最大值）
    15-2. 若為小於 40'hFF80000000，mt_temp40 後 32 位數更新為 32'h80000000（32 位有號數最小值）
    15-3. mt_temp40 後 32 位數賦值給 result，前 8 位數賦值給 protect
16. 若為 5，mt_er8_pre & mt_cand8_pre 相乘賦值給 mt_temp20_pre，其後

16 位數賦值給 result 後 16 位數，前 4 位數賦值給 protect 後 4 位數；而 mt_er8_last & mt_cand8_last 相乘賦值給 mt_temp20_last，其後 16 位數賦值給 result 前 16 位數，前 4 位數賦值給 protect 前 4 位數

17. mt_er8_pre & mt_cand8_pre 相乘加上原本的 mt_temp20_pre 更新 mt_temp20_pre ，其後 16 位數賦值給 result 後 16 位數，前 4 位數賦值給 protect 後 4 位數；而 mt_er8_last & mt_cand8_last 相乘加上原本的 mt_temp20_last 更新 mt_temp20_last，其後 16 位數賦值給 result 前 16 位數，前 4 位數賦值給 protect 前 4 位數

18. 若為 7，接著分別判斷 mt_temp20_pre 及 mt_temp20_last 的值:

    18-1./ 18-3. 若為大於 20'h07FFF，mt_temp20_pre/ _last 後 16 位數更新為 16'h7FFF（16 位有號數最大值）

    18-2./ 18-4. 若為小於 20'hF8000，mt_temp20_pre/ _last 後 16 位數更新為 16'h8000（16 位有號數最小值）

    18-5. mt_temp20_pre 後 16 位數賦值給 result 後 16 位數，前 4 位數賦值給 protect 後 4 位數；而 mt_temp20_last 後 16 位數賦值給 result 前 16 位數，前 4 位數賦值給 protect 前 4 位數