晶片安全設計

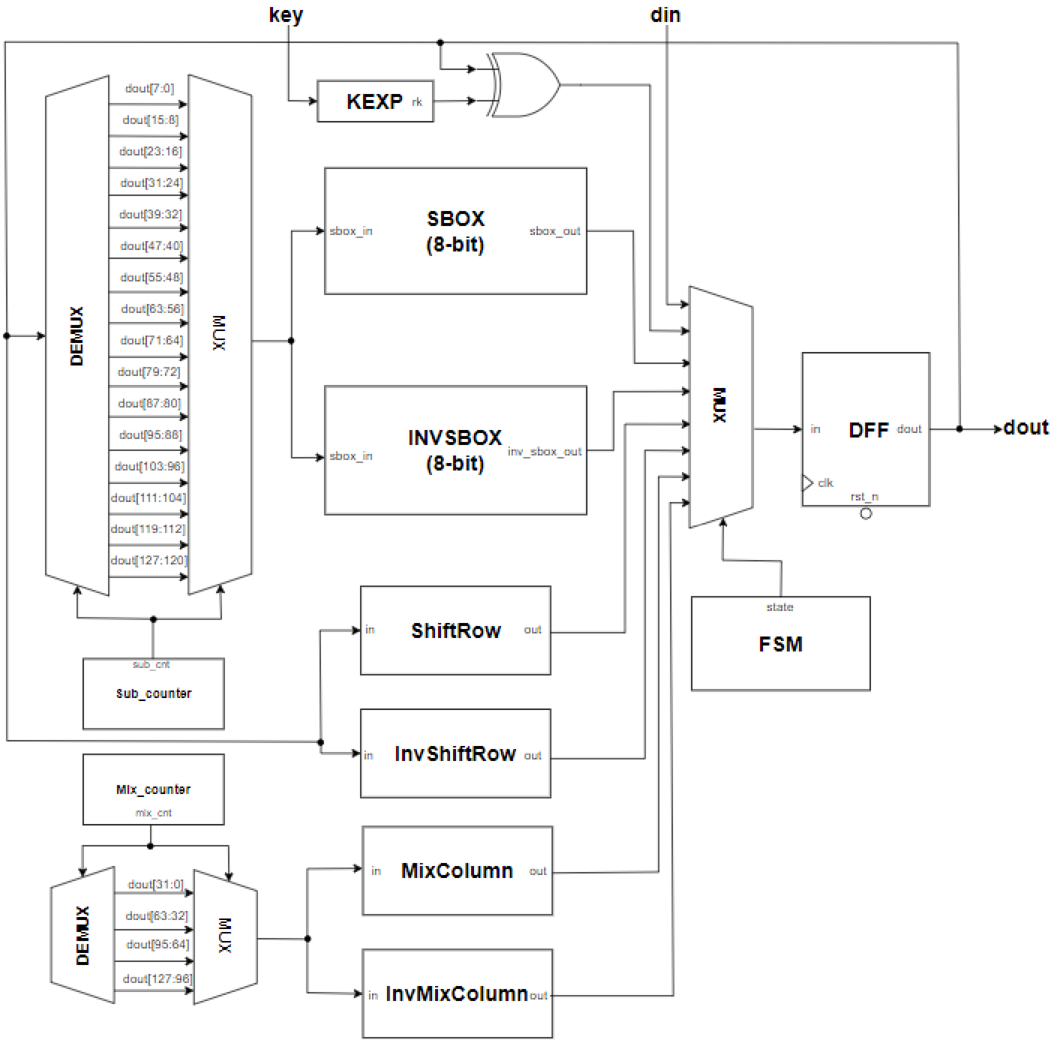
Lab03

姓名:王證皓

學號:112061619

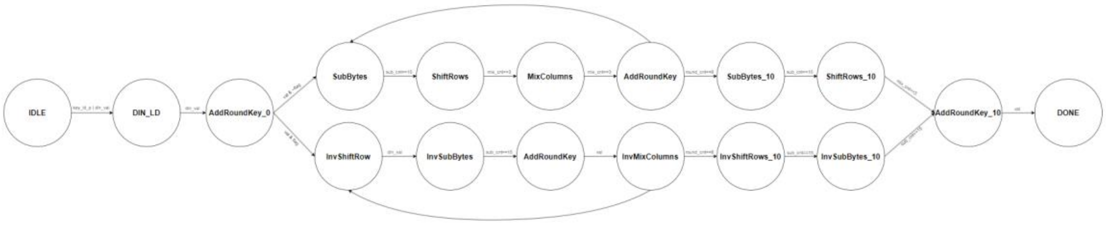
一、 Architecture

1. Block Diagram



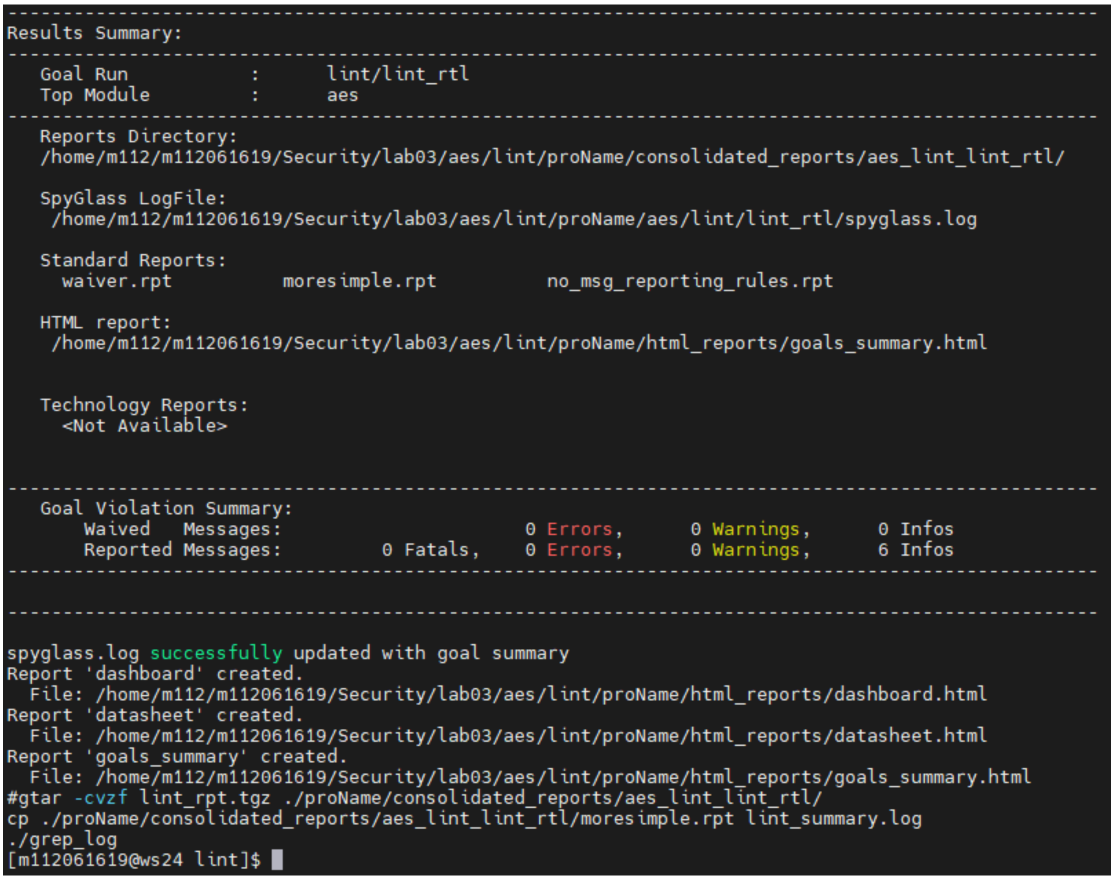
電路架構一開始會先將 key 輸入至 kexp 模組存起來，接著將 key 做擴展成 11 筆 key，並且在不同的 round 中輸出對應的 key，din 一開始則會透過 DFF 先將資料存至 dout，接著 dout 會拉回至左邊的 SBOX, INVSBOX, ShiftRow, InvShiftRow, MixColumn, InvColumn 做運算，並根據不同的 state 使用 MUX 來選擇如何更新 dout 值，其中 SBOX 因為只使用 8-bit，所以會需 sub_counter 以及 DEMUX, MUX 來做為選擇做為 SBOX 的 input，MixColumn 也是相同的做法，只是 MixColumn 使用的是 32-bit，所以只有四種輸入。

2. FSM



FSM 在 IDLE state 做 reset 後進到 DIN_LD，此 state 須等到接收到 Din 才會進入 AddRoudKey_0，接著根據是加密或解密模式選擇上或下的 path，加密模式下會經 SubBytes, ShiftRows, MixColumns, AddRoundKey，並重複幾次後，進入最後一輪的 SubBytes_10,ShiftRows_10, AddRoundKey_10 最後進入 DONE，解密的話也是相同，只是操作的方法為反向操作，其中 AddRoundKey 會須等到正確接收 round key 才會進入下一 state，SubBytes 則是會等待數 15 次的 counter，MixColumns 則是會等 4 個 cycle。

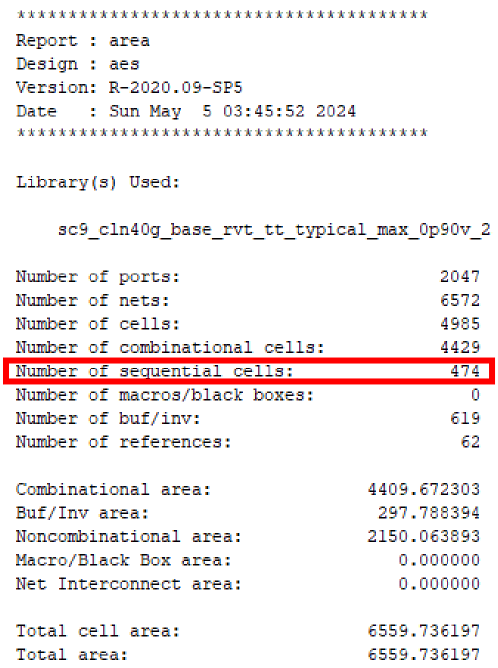
3. Lint Check



```
------------------------------------------------------------------------------------------------------
Results Summary:
------------------------------------------------------------------------------------------------------
   Goal Run           :      lint/lint_rtl
   Top Module         :      aes
------------------------------------------------------------------------------------------------------
   Reports Directory:
   /home/m112/m112061619/Security/lab03/aes/lint/proName/consolidated_reports/aes_lint_lint_rtl/

   SpyGlass LogFile:
    /home/m112/m112061619/Security/lab03/aes/lint/proName/aes/lint/lint_rtl/spyglass.log

   Standard Reports:
     waiver.rpt          moresimple.rpt          no_msg_reporting_rules.rpt

   HTML report:
    /home/m112/m112061619/Security/lab03/aes/lint/proName/html_reports/goals_summary.html


   Technology Reports:
     <Not Available>


------------------------------------------------------------------------------------------------------
   Goal Violation Summary:
       Waived   Messages:                    0 Errors,       0 Warnings,       0 Infos
       Reported Messages:         0 Fatals,  0 Errors,       0 Warnings,       6 Infos
------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------

spyglass.log successfully updated with goal summary
Report 'dashboard' created.
  File: /home/m112/m112061619/Security/lab03/aes/lint/proName/html_reports/dashboard.html
Report 'datasheet' created.
  File: /home/m112/m112061619/Security/lab03/aes/lint/proName/html_reports/datasheet.html
Report 'goals_summary' created.
  File: /home/m112/m112061619/Security/lab03/aes/lint/proName/html_reports/goals_summary.html
#gtar -cvzf lint_rpt.tgz ./proName/consolidated_reports/aes_lint_lint_rtl/
cp ./proName/consolidated_reports/aes_lint_lint_rtl/moresimple.rpt lint_summary.log
./grep_log
[m112061619@ws24 lint]$ █
```

二、 Resource

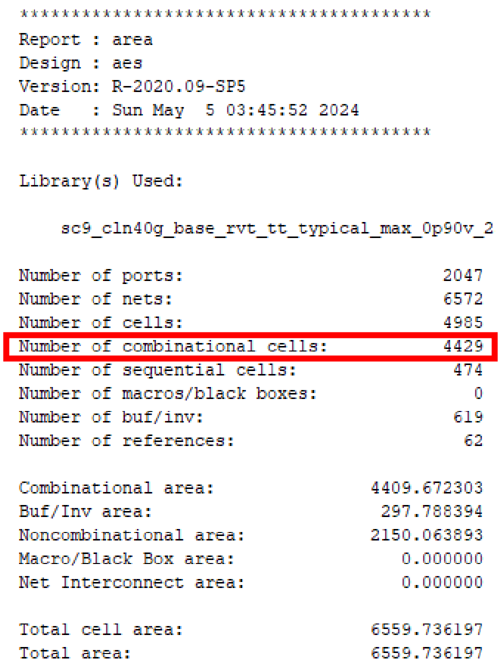
1. Flip-Flop



```
****************************************
Report : area
Design : aes
Version: R-2020.09-SP5
Date   : Sun May  5 03:45:52 2024
****************************************

Library(s) Used:

    sc9_cln40g_base_rvt_tt_typical_max_0p90v_2

Number of ports:                         2047
Number of nets:                          6572
Number of cells:                         4985
Number of combinational cells:           4429
Number of sequential cells:               474
Number of macros/black boxes:               0
Number of buf/inv:                        619
Number of references:                      62

Combinational area:               4409.672303
Buf/Inv area:                      297.788394
Noncombinational area:            2150.063893
Macro/Black Box area:                0.000000
Net Interconnect area:               0.000000

Total cell area:                  6559.736197
Total area:                       6559.736197
```

Number of FF:474

2. Combinational



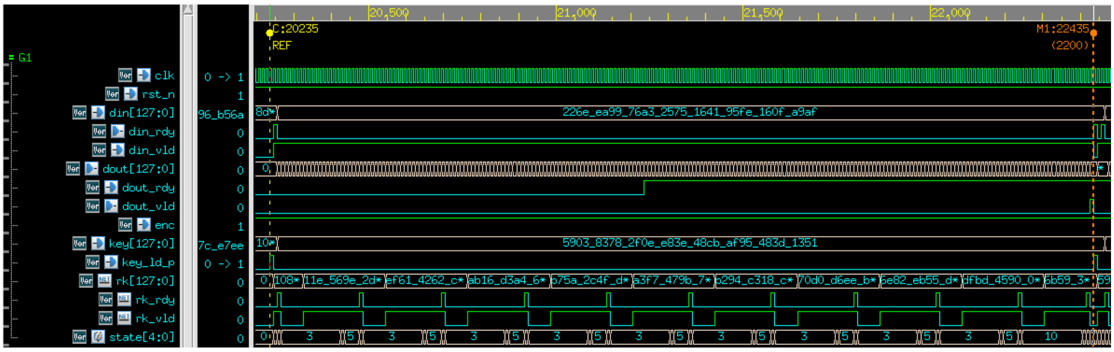
```
****************************************
Report : area
Design : aes
Version: R-2020.09-SP5
Date   : Sun May  5 03:45:52 2024
****************************************

Library(s) Used:

    sc9_cln40g_base_rvt_tt_typical_max_0p90v_2

Number of ports:                         2047
Number of nets:                          6572
Number of cells:                         4985
Number of combinational cells:           4429
Number of sequential cells:               474
Number of macros/black boxes:               0
Number of buf/inv:                        619
Number of references:                      62

Combinational area:               4409.672303
Buf/Inv area:                      297.788394
Noncombinational area:            2150.063893
Macro/Black Box area:                0.000000
Net Interconnect area:               0.000000

Total cell area:                  6559.736197
Total area:                       6559.736197
```

Combinational Operations:4429

Total area:6559

三、 Performance

1. Latency and Throughput(enc)

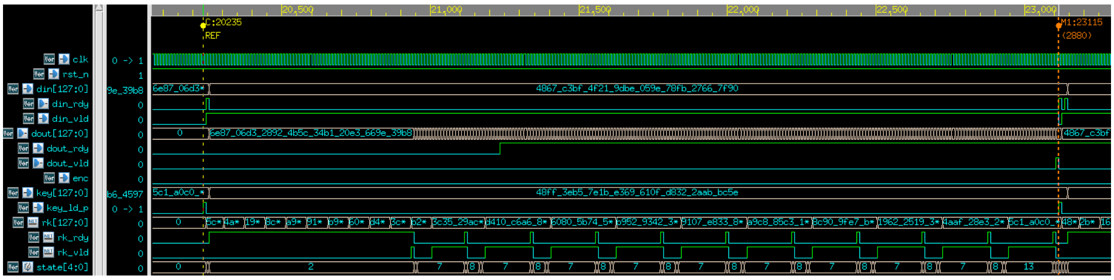


Latency:2200 ns(2200/10=220 cycle)

Throughput:1/220=0.00455 筆/cycle

=128bit/2200ns=58.18Mbps

2. Latency and Throughput(dec)



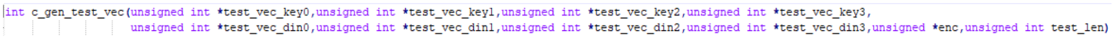
Latency:2880 ns(2880/10=288 cycle)

Throughput:1/288=0.00347 筆/cycle

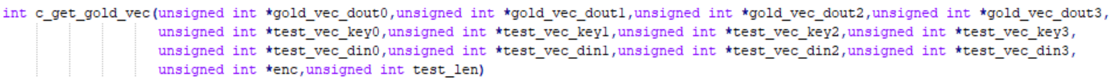
=128bit/2880ns=44.44Mbps

四、 BFM Testbench

1. C_Model



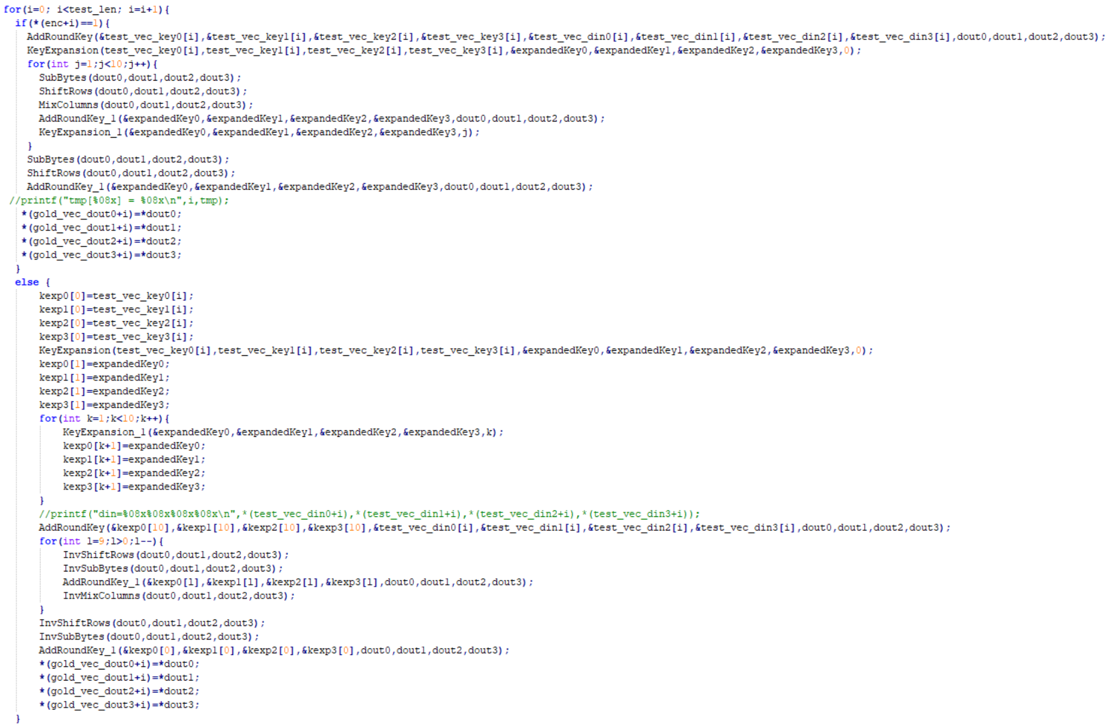
```
int c_gen_test_vec(unsigned int *test_vec_key0,unsigned int *test_vec_key1,unsigned int *test_vec_key2,unsigned int *test_vec_key3,
                   unsigned int *test_vec_din0,unsigned int *test_vec_din1,unsigned int *test_vec_din2,unsigned int *test_vec_din3,unsigned *enc,unsigned int test_len)
```



```
int c_get_gold_vec(unsigned int *gold_vec_dout0,unsigned int *gold_vec_dout1,unsigned int *gold_vec_dout2,unsigned int *gold_vec_dout3,
                   unsigned int *test_vec_key0,unsigned int *test_vec_key1,unsigned int *test_vec_key2,unsigned int *test_vec_key3,
                   unsigned int *test_vec_din0,unsigned int *test_vec_din1,unsigned int *test_vec_din2,unsigned int *test_vec_din3,
                   unsigned int *enc,unsigned int test_len)
```

上圖有兩個 function，c_gen_test_vec 為產生輸入 pattern 的函數，輸出有 key 與 din，c_gat_gold_vec 為產生 golden data 的函數。

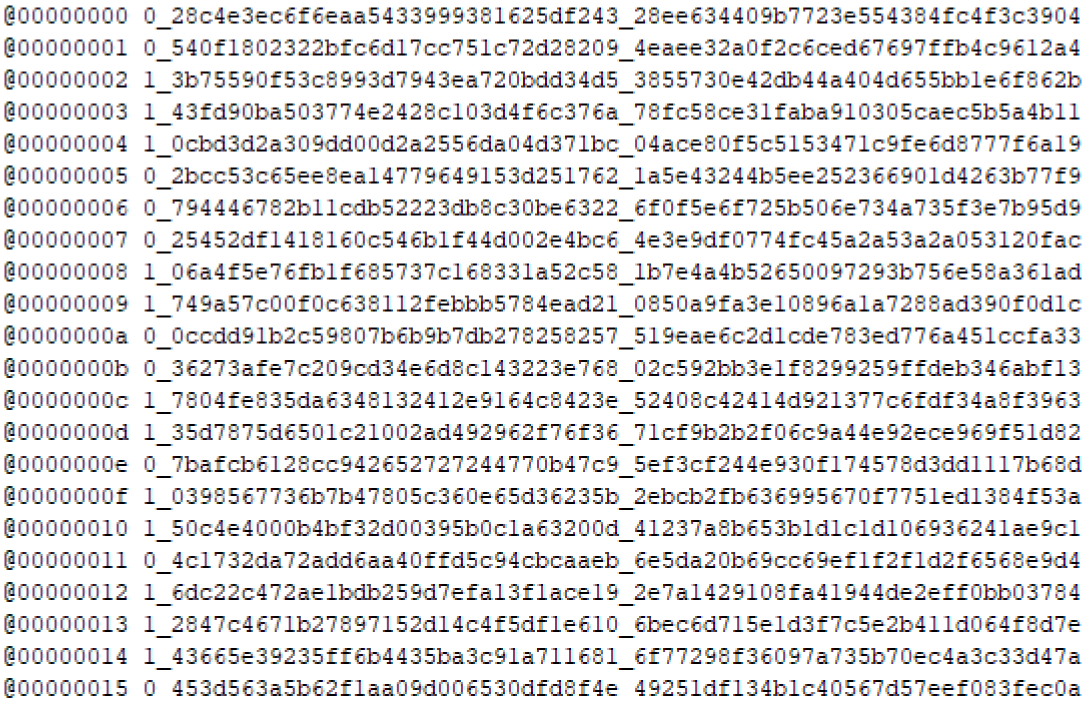
2. test_vector_gen



```
for(i=0; i<test_len; i=i+1){
  if(*(enc+i)==1){
    AddRoundKey(&test_vec_key0[i],&test_vec_key1[i],&test_vec_key2[i],&test_vec_key3[i],&test_vec_din0[i],&test_vec_din1[i],&test_vec_din2[i],&test_vec_din3[i],dout0,dout1,dout2,dout3);
    KeyExpansion(test_vec_key0[i],test_vec_key1[i],test_vec_key2[i],test_vec_key3[i],&expandedKey0,&expandedKey1,&expandedKey2,&expandedKey3,0);
    for(int j=1;j<10;j++){
      SubBytes(dout0,dout1,dout2,dout3);
      ShiftRows(dout0,dout1,dout2,dout3);
      MixColumns(dout0,dout1,dout2,dout3);
      AddRoundKey_1(&expandedKey0,&expandedKey1,&expandedKey2,&expandedKey3,dout0,dout1,dout2,dout3);
      KeyExpansion_1(&expandedKey0,&expandedKey1,&expandedKey2,&expandedKey3,j);
    }
    SubBytes(dout0,dout1,dout2,dout3);
    ShiftRows(dout0,dout1,dout2,dout3);
    AddRoundKey_1(&expandedKey0,&expandedKey1,&expandedKey2,&expandedKey3,dout0,dout1,dout2,dout3);
 //printf("tmp[%08x] = %08x\n",i,tmp);
   *(gold_vec_dout0+i)=*dout0;
   *(gold_vec_dout1+i)=*dout1;
   *(gold_vec_dout2+i)=*dout2;
   *(gold_vec_dout3+i)=*dout3;
  }
  else {
      kexp0[0]=test_vec_key0[i];
      kexp1[0]=test_vec_key1[i];
      kexp2[0]=test_vec_key2[i];
      kexp3[0]=test_vec_key3[i];
      KeyExpansion(test_vec_key0[i],test_vec_key1[i],test_vec_key2[i],test_vec_key3[i],&expandedKey0,&expandedKey1,&expandedKey2,&expandedKey3,0);
      kexp0[1]=expandedKey0;
      kexp1[1]=expandedKey1;
      kexp2[1]=expandedKey2;
      kexp3[1]=expandedKey3;
      for(int k=1;k<10;k++){
          KeyExpansion_1(&expandedKey0,&expandedKey1,&expandedKey2,&expandedKey3,k);
          kexp0[k+1]=expandedKey0;
          kexp1[k+1]=expandedKey1;
          kexp2[k+1]=expandedKey2;
          kexp3[k+1]=expandedKey3;
      }
      //printf("din=%08x%08x%08x%08x\n",*(test_vec_din0+i),*(test_vec_din1+i),*(test_vec_din2+i),*(test_vec_din3+i));
      AddRoundKey(&kexp0[10],&kexp1[10],&kexp2[10],&kexp3[10],&test_vec_din0[i],&test_vec_din1[i],&test_vec_din2[i],&test_vec_din3[i],dout0,dout1,dout2,dout3);
      for(int l=9;l>0;l--){
          InvShiftRows(dout0,dout1,dout2,dout3);
          InvSubBytes(dout0,dout1,dout2,dout3);
          AddRoundKey_1(&kexp0[l],&kexp1[l],&kexp2[l],&kexp3[l],dout0,dout1,dout2,dout3);
          InvMixColumns(dout0,dout1,dout2,dout3);
      }
      InvShiftRows(dout0,dout1,dout2,dout3);
      InvSubBytes(dout0,dout1,dout2,dout3);
      AddRoundKey_1(&kexp0[0],&kexp1[0],&kexp2[0],&kexp3[0],dout0,dout1,dout2,dout3);
      *(gold_vec_dout0+i)=*dout0;
      *(gold_vec_dout1+i)=*dout1;
      *(gold_vec_dout2+i)=*dout2;
      *(gold_vec_dout3+i)=*dout3;
  }
```

上圖為使用前面所述的兩個函數，根據目前是加密或解密模式透過 pointer 將 data 讀出並寫入 dat 檔案。

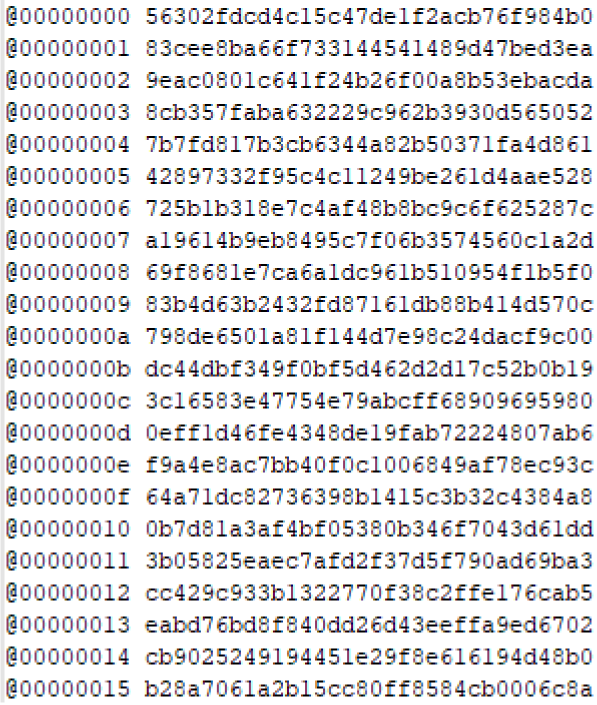
3. Test Vector



```
@00000000 0_28c4e3ec6f6eaa5433999381625df243_28ee634409b7723e554384fc4f3c3904
@00000001 0_540f1802322bfc6d17cc751c72d28209_4eaee32a0f2c6ced67697ffb4c9612a4
@00000002 1_3b75590f53c8993d7943ea720bdd34d5_3855730e42db44a404d655bb1e6f862b
@00000003 1_43fd90ba503774e2428c103d4f6c376a_78fc58ce31faba910305caec5b5a4b11
@00000004 1_0cbd3d2a309dd00d2a2556da04d371bc_04ace80f5c5153471c9fe6d8777f6a19
@00000005 0_2bcc53c65ee8ea14779649153d251762_1a5e43244b5ee252366901d4263b77f9
@00000006 0_794446782b11cdb52223db8c30be6322_6f0f5e6f725b506e734a735f3e7b95d9
@00000007 0_25452df1418160c546b1f44d002e4bc6_4e3e9df0774fc45a2a53a2a053120fac
@00000008 1_06a4f5e76fb1f685737c168331a52c58_1b7e4a4b52650097293b756e58a361ad
@00000009 1_749a57c00f0c638112febbb5784ead21_0850a9fa3e10896a1a7288ad390f0d1c
@0000000a 0_0ccdd91b2c59807b6b9b7db278258257_519eae6c2d1cde783ed776a451ccfa33
@0000000b 0_36273afe7c209cd34e6d8c143223e768_02c592bb3e1f8299259ffdeb346abf13
@0000000c 1_7804fe835da6348132412e9164c8423e_52408c42414d921377c6fdf34a8f3963
@0000000d 1_35d7875d6501c21002ad492962f76f36_71cf9b2b2f06c9a44e92ece969f51d82
@0000000e 0_7bafcb6128cc942652727244770b47c9_5ef3cf244e930f174578d3dd1117b68d
@0000000f 1_0398567736b7b47805c360e65d36235b_2ebcb2fb636995670f7751ed1384f53a
@00000010 1_50c4e4000b4bf32d00395b0c1a63200d_41237a8b653b1d1c1d106936241ae9c1
@00000011 0_4c1732da72add6aa40ffd5c94cbcaaeb_6e5da20b69cc69ef1f2f1d2f6568e9d4
@00000012 1_6dc22c472ae1bdb259d7efa13f1ace19_2e7a1429108fa41944de2eff0bb03784
@00000013 1_2847c4671b27897152d14c4f5df1e610_6bec6d715e1d3f7c5e2b411d064f8d7e
@00000014 1_43665e39235ff6b4435ba3c91a711681_6f77298f36097a735b70ec4a3c33d47a
@00000015 0 453d563a5b62f1aa09d006530dfd8f4e 49251df134b1c40567d57eef083fec0a
```

最左邊的bit代表enc的數值，中間的128 bit代表key的數值，最右邊則為din的數值。

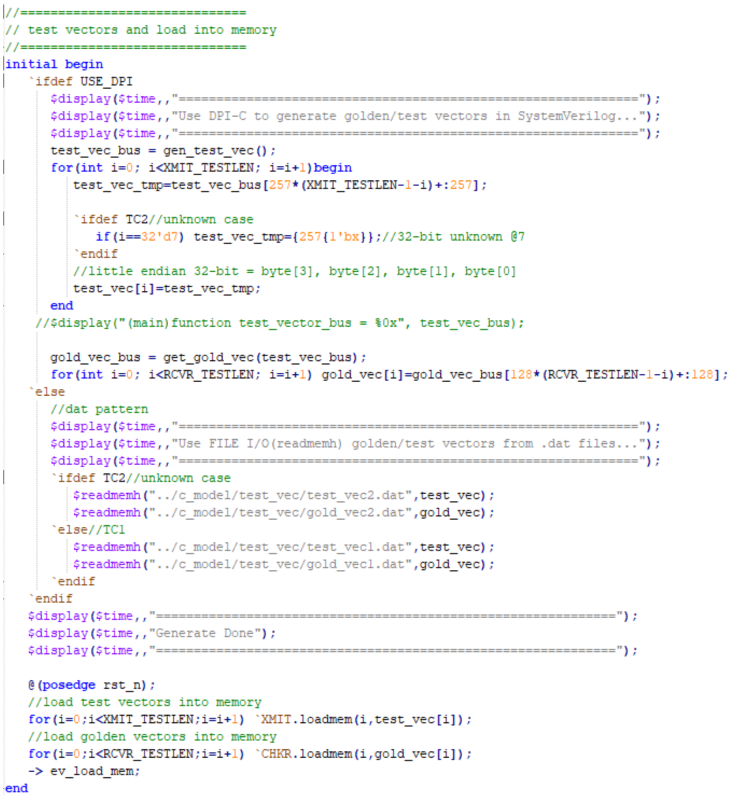
4. Golden Data



```
@00000000 56302fdcd4c15c47de1f2acb76f984b0
@00000001 83cee8ba66f73314454148 9d47bed3ea
@00000002 9eac0801c641f24b26f00a8b53ebacda
@00000003 8cb357faba632229c962b3930d565052
@00000004 7b7fd817b3cb6344a82b50371fa4d861
@00000005 42897332f95c4c11249be261d4aae528
@00000006 725b1b318e7c4af48b8bc9c6f625287c
@00000007 a19614b9eb8495c7f06b3574560c1a2d
@00000008 69f8681e7ca6a1dc961b510954f1b5f0
@00000009 83b4d63b2432fd87161db88b414d570c
@0000000a 798de6501a81f144d7e98c24dacf9c00
@0000000b dc44dbf349f0bf5d462d2d17c52b0b19
@0000000c 3c16583e47754e79abcff68909695980
@0000000d 0eff1d46fe4348de19fab72224807ab6
@0000000e f9a4e8ac7bb40f0c1006849af78ec93c
@0000000f 64a71dc82736398b1415c3b32c4384a8
@00000010 0b7d81a3af4bf05380b346f7043d61dd
@00000011 3b05825eaec7afd2f37d5f790ad69ba3
@00000012 cc429c933b1322770f38c2ffe176cab5
@00000013 eabd76bd8f840dd26d43eeffa9ed6702
@00000014 cb902524919445 1e29f8e616194d48b0
@00000015 b28a7061a2b15cc80ff8584cb0006c8a
```

上圖為經加密或解密出來的golden data

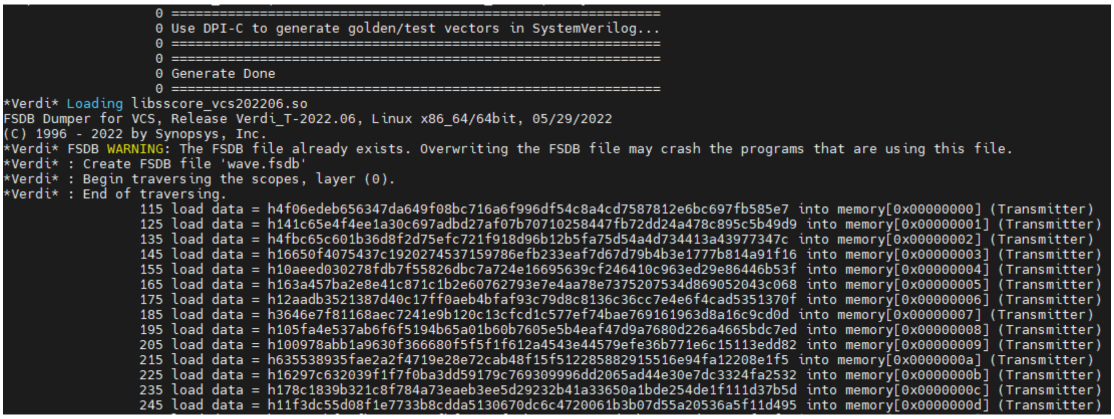
5. DPI-C & FILE I/O



```
//==============================
// test vectors and load into memory
//==============================
initial begin
   `ifdef USE_DPI
     $display($time,,"============================================================");
     $display($time,,"Use DPI-C to generate golden/test vectors in SystemVerilog...");
     $display($time,,"============================================================");
     test_vec_bus = gen_test_vec();
     for(int i=0; i<XMIT_TESTLEN; i=i+1)begin
        test_vec_tmp=test_vec_bus[257*(XMIT_TESTLEN-1-i)+:257];

        `ifdef TC2//unknown case
           if(i==32'd7) test_vec_tmp={257{1'bx}};//32-bit unknown @7
        `endif
        //little endian 32-bit = byte[3], byte[2], byte[1], byte[0]
        test_vec[i]=test_vec_tmp;
     end
    //$display("(main)function test_vector_bus = %0x", test_vec_bus);

     gold_vec_bus = get_gold_vec(test_vec_bus);
     for(int i=0; i<RCVR_TESTLEN; i=i+1) gold_vec[i]=gold_vec_bus[128*(RCVR_TESTLEN-1-i)+:128];
   `else
     //dat pattern
     $display($time,,"============================================================");
     $display($time,,"Use FILE I/O(readmemh) golden/test vectors from .dat files...");
     $display($time,,"============================================================");
     `ifdef TC2//unknown case
        $readmemh("../c_model/test_vec/test_vec2.dat",test_vec);
        $readmemh("../c_model/test_vec/gold_vec2.dat",gold_vec);
     `else//TC1
        $readmemh("../c_model/test_vec/test_vec1.dat",test_vec);
        $readmemh("../c_model/test_vec/gold_vec1.dat",gold_vec);
     `endif
   `endif
   $display($time,,"============================================================");
   $display($time,,"Generate Done");
   $display($time,,"============================================================");

   @(posedge rst_n);
   //load test vectors into memory
   for(i=0;i<XMIT_TESTLEN;i=i+1) `XMIT.loadmem(i,test_vec[i]);
   //load golden vectors into memory
   for(i=0;i<RCVR_TESTLEN;i=i+1) `CHKR.loadmem(i,gold_vec[i]);
   -> ev_load_mem;
end
```

此部分為 testbench，其中能選擇使用 DPI-C 或是 FILE I/O 模式，DPI-C 是透過前面的 C function 將資料轉換成 system-verilog 能使用的格式，FILE I/O 則是使用前面所產生的 pattern。

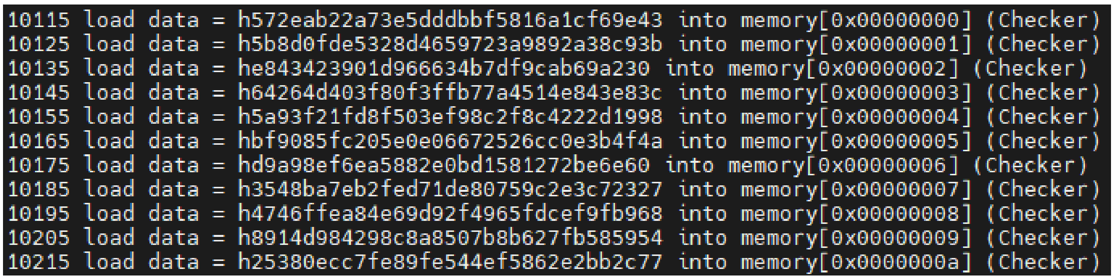
6. Bfm_xmit



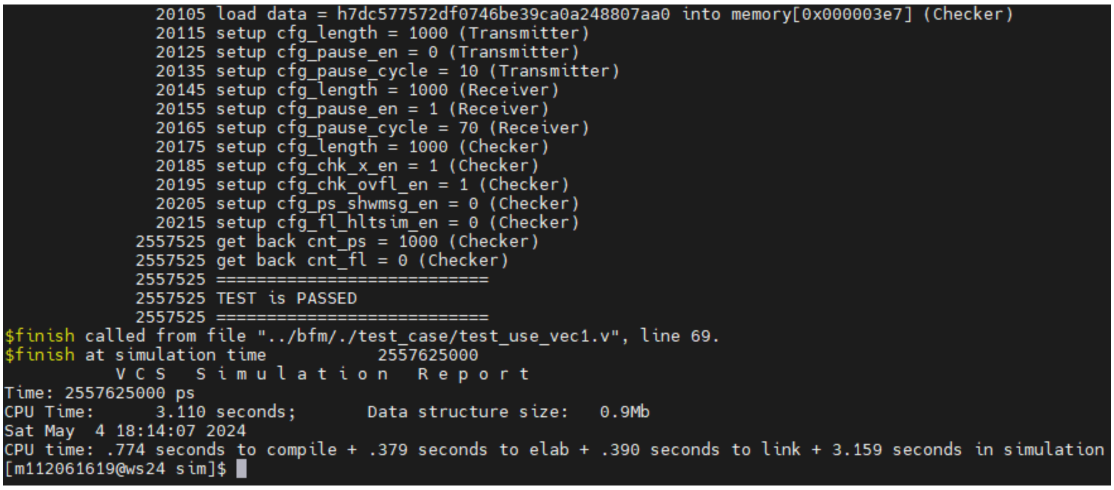
```
                   0 ==============================================================
                   0 Use DPI-C to generate golden/test vectors in SystemVerilog...
                   0 ==============================================================
                   0 ==============================================================
                   0 Generate Done
                   0 ==============================================================
*Verdi* Loading libsscore_vcs202206.so
FSDB Dumper for VCS, Release Verdi_T-2022.06, Linux x86_64/64bit, 05/29/2022
(C) 1996 - 2022 by Synopsys, Inc.
*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 'wave.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
                 115 load data = h4f06edeb656347da649f08bc716a6f996df54c8a4cd7587812e6bc697fb585e7 into memory[0x00000000] (Transmitter)
                 125 load data = h141c65e4f4ee1a30c697adbd27af07b70710258447fb72dd24a478c895c5b49d9 into memory[0x00000001] (Transmitter)
                 135 load data = h4fbc65c601b36d8f2d75efc721f918d96b12b5fa75d54a4d734413a43977347c into memory[0x00000002] (Transmitter)
                 145 load data = h16650f4075437c1920274537159786efb233eaf7d67d79b4b3e1777b814a91f16 into memory[0x00000003] (Transmitter)
                 155 load data = h10aeed030278fdb7f55826dbc7a724e16695639cf246410c963ed29e86446b53f into memory[0x00000004] (Transmitter)
                 165 load data = h163a457ba2e8e41c871c1b2e60762793e7e4aa78e7375207534d869052043c068 into memory[0x00000005] (Transmitter)
                 175 load data = h12aadb3521387d40c17ff0aeb4bfaf93c79d8c8136c36cc7e4e6f4cad5351370f into memory[0x00000006] (Transmitter)
                 185 load data = h3646e7f81168aec7241e9b120c13cfcd1c577ef74bae769161963d8a16c9cd0d into memory[0x00000007] (Transmitter)
                 195 load data = h105fa4e537ab6f6f5194b65a01b60b7605e5b4eaf47d9a7680d226a4665bdc7ed into memory[0x00000008] (Transmitter)
                 205 load data = h100978abb1a9630f366680f5f5f1f612a4543e44579efe36b771e6c15113edd82 into memory[0x00000009] (Transmitter)
                 215 load data = h635538935fae2a2f4719e28e72cab48f15f51228588291551 6e94fa12208e1f5 into memory[0x0000000a] (Transmitter)
                 225 load data = h16297c632039f1f7f0ba3dd59179c769309996dd2065ad44e30e7dc3324fa2532 into memory[0x0000000b] (Transmitter)
                 235 load data = h178c1839b321c8f784a73eaeb3ee5d29232b41a33650a1bde254de1f111d37b5d into memory[0x0000000c] (Transmitter)
                 245 load data = h11f3dc55d08f1e7733b8cdda5130670dc6c4720061b3b07d55a20536a5f11d495 into memory[0x0000000d] (Transmitter)
```

Transmitter 將資料寫進 memory 裡，並透過 din 的 protocol 輸入至 aes 模組中。

7. Bfm_rcvr & Bfm_chkr



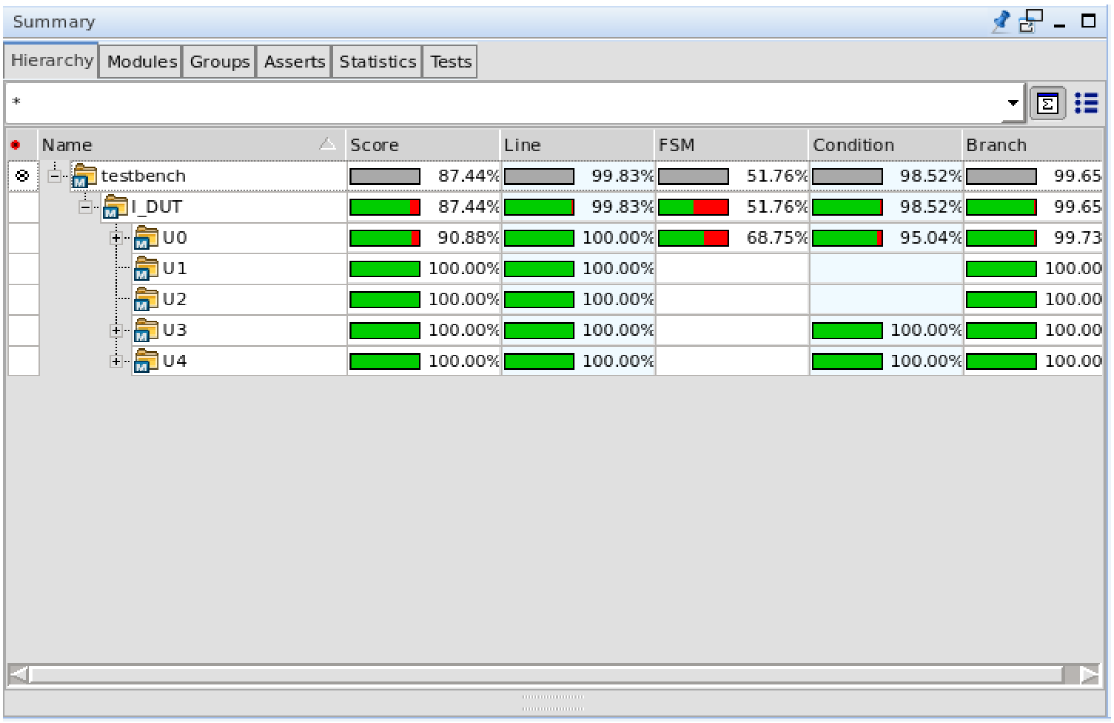
```
10115 load data = h572eab22a73e5dddbbf5816a1cf69e43 into memory[0x00000000] (Checker)
10125 load data = h5b8d0fde5328d4659723a9892a38c93b into memory[0x00000001] (Checker)
10135 load data = he843423901d966634b7df9cab69a230 into memory[0x00000002] (Checker)
10145 load data = h64264d403f80f3ffb77a4514e843e83c into memory[0x00000003] (Checker)
10155 load data = h5a93f21fd8f503ef98c2f8c4222d1998 into memory[0x00000004] (Checker)
10165 load data = hbf9085fc205e0e06672526cc0e3b4f4a into memory[0x00000005] (Checker)
10175 load data = hd9a98ef6ea5882e0bd1581272be6e60 into memory[0x00000006] (Checker)
10185 load data = h3548ba7eb2fed71de80759c2e3c72327 into memory[0x00000007] (Checker)
10195 load data = h4746ffea84e69d92f4965fdcef9fb968 into memory[0x00000008] (Checker)
10205 load data = h8914d984298c8a8507b8b627fb585954 into memory[0x00000009] (Checker)
10215 load data = h25380ecc7fe89fe544ef5862e2bb2c77 into memory[0x0000000a] (Checker)
```



```
               20105 load data = h7dc577572df0746be39ca0a248807aa0 into memory[0x000003e7] (Checker)
               20115 setup cfg_length = 1000 (Transmitter)
               20125 setup cfg_pause_en = 0 (Transmitter)
               20135 setup cfg_pause_cycle = 10 (Transmitter)
               20145 setup cfg_length = 1000 (Receiver)
               20155 setup cfg_pause_en = 1 (Receiver)
               20165 setup cfg_pause_cycle = 70 (Receiver)
               20175 setup cfg_length = 1000 (Checker)
               20185 setup cfg_chk_x_en = 1 (Checker)
               20195 setup cfg_chk_ovfl_en = 1 (Checker)
               20205 setup cfg_ps_shwmsg_en = 0 (Checker)
               20215 setup cfg_fl_hltsim_en = 0 (Checker)
             2557525 get back cnt_ps = 1000 (Checker)
             2557525 get back cnt_fl = 0 (Checker)
             2557525 ===========================
             2557525 TEST is PASSED
             2557525 ===========================
$finish called from file "../bfm/./test_case/test_use_vec1.v", line 69.
$finish at simulation time           2557625000
           V C S   S i m u l a t i o n   R e p o r t
Time: 2557625000 ps
CPU Time:      3.110 seconds;       Data structure size:   0.9Mb
Sat May  4 18:14:07 2024
CPU time: .774 seconds to compile + .379 seconds to elab + .390 seconds to link + 3.159 seconds in simulation
[m112061619@ws24 sim]$ 
```

使用 Receiver 並透過 dout 的 protocol 將資料接收後，使用 Checker 檢查比對資料是否與 golden data 相符。

五、 Code Coverage

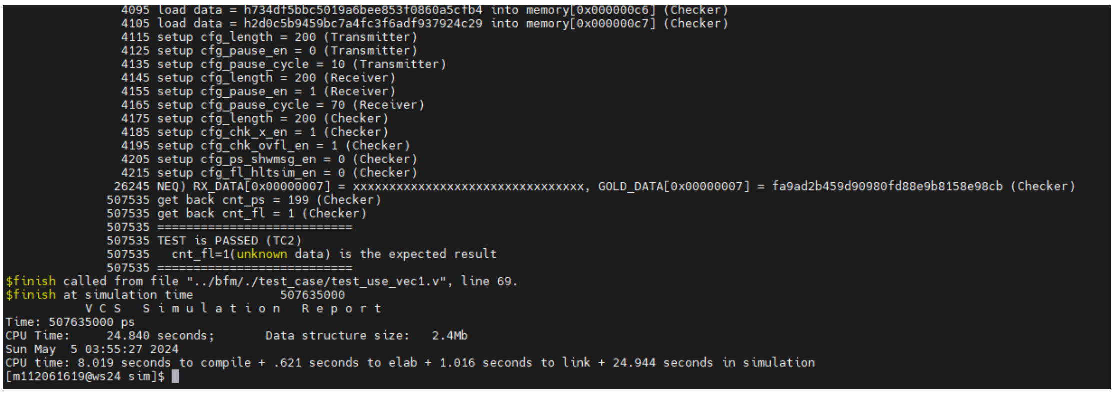


Name	Score	Line	FSM	Condition	Branch
testbench	87.44%	99.83%	51.76%	98.52%	99.65
I_DUT	87.44%	99.83%	51.76%	98.52%	99.65
U0	90.88%	100.00%	68.75%	95.04%	99.73
U1	100.00%	100.00%			100.00
U2	100.00%	100.00%			100.00
U3	100.00%	100.00%		100.00%	100.00
U4	100.00%	100.00%		100.00%	100.00

Code Coverage 的部分在 bfm 中，使用 5 組 test case 做測試，分別為調整 tx

與 rx 其 valid 的時間並加入 overflow 的 case，可得到以上每種 coverage 皆接近 100，唯獨 FSM 的 coverage 較低，推測因我有加入 unknown 的 pattern(xxxx)，導致 state 無法做轉換，所以導致 FSM coverage 較低。

六、 Gate-Level Simulation



```
               4095 load data = h734df5bbc5019a6bee853f0860a5cfb4 into memory[0x000000c6] (Checker)
               4105 load data = h2d0c5b9459bc7a4fc3f6adf937924c29 into memory[0x000000c7] (Checker)
               4115 setup cfg_length = 200 (Transmitter)
               4125 setup cfg_pause_en = 0 (Transmitter)
               4135 setup cfg_pause_cycle = 10 (Transmitter)
               4145 setup cfg_length = 200 (Receiver)
               4155 setup cfg_pause_en = 1 (Receiver)
               4165 setup cfg_pause_cycle = 70 (Receiver)
               4175 setup cfg_length = 200 (Checker)
               4185 setup cfg_chk_x_en = 1 (Checker)
               4195 setup cfg_chk_ovfl_en = 1 (Checker)
               4205 setup cfg_ps_shwmsg_en = 0 (Checker)
               4215 setup cfg_fl_hltsim_en = 0 (Checker)
              26245 NEQ) RX_DATA[0x00000007] = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, GOLD_DATA[0x00000007] = fa9ad2b459d90980fd88e9b8158e98cb (Checker)
             507535 get back cnt_ps = 199 (Checker)
             507535 get back cnt_fl = 1 (Checker)
             507535 ===========================
             507535 TEST is PASSED (TC2)
             507535   cnt_fl=1(unknown data) is the expected result
             507535 ===========================
$finish called from file "../bfm/./test_case/test_use_vec1.v", line 69.
$finish at simulation time            507635000
           V C S   S i m u l a t i o n   R e p o r t
Time: 507635000 ps
CPU Time:     24.840 seconds;       Data structure size:   2.4Mb
Sun May  5 03:55:27 2024
CPU time: 8.019 seconds to compile + .621 seconds to elab + 1.016 seconds to link + 24.944 seconds in simulation
[m112061619@ws24 sim]$ █
```