# 國立清華大學

# 超大型積體電路測試

# VLSI Testing


國立清華大學

NATIONAL TSING HUA UNIVERSITY

## Homework 2

系所級:電子所一年級、電機系碩士班一年級

學號:111063548、111061624

姓名:蕭方凱、尤弘瑋

指導老師:黃錫瑜教授

# 目錄

**(a) Write the RTL code in Verilog or VHDL that takes in two 8-bit positive integers, A[7:0] and B[7:0], and produces its quotient Q[7:0] and remainder R[7:0]. ..... 4**

divider.v ..... 4

Finite State Machine ..... 6

**(b) Verify the correctness of your RTL code by a testbench. You should try it out by at least 3 pairs of input numbers. ..... 7**

PATTERN.v ..... 7

TESTBED.v ..... 12

Display on MobaXterm ..... 13

nWave ..... 13

**(c) Use a synthesis script to convert your RTL code into a gate-level netlist. Report the final gate count, the maximum operating speed (in MHz) and the estimated power dissipation in (mW) using Design Compiler. ..... 14**

Area Report ..... 14

Slack Report ..... 15

Power dissipation ..... 16

**(d) Add the scan chain into your gate-level netlist obtained by part(c), report the resulting gate count, the maximum operating speed (in MHz) of your circuit. Compare to the non-scan version, and report the area overhead percentage and performance penalty de to scan chain insertion. ..... 17**

Area Report ..... 18

Slack Report ..... 19

Power dissipation ..... 20

Comparing before/after scan chain ..... 20

Comment: ..... 21

**(e) Run ATPG using a commercial tool available and report the fault coverage. ..... 22**

# 圖目錄

FIG 1 RTL CODE(DIVIDER.V) ..... 4
FIG 2 RTL CODE(DIVIDER.V) ..... 5
FIG 3 PATTERN.V (I/O PORTS AND PARAMETERS) ..... 7
FIG 4 PATTERN.V (MAIN TASK STRUCTURE) ..... 8
FIG 5 PATTERN.V (CLOCK GENERATION) ..... 8
FIG 6 PATTERN.V (RESET TASK) ..... 9
FIG 7 PATTERN.V (INPUT TASK) ..... 9
FIG 8 PATTERN.V (CHECK TASK) ..... 9
FIG 9 PATTERN.V (ERROR TASK) ..... 9
FIG 10 PATTERN.V (FAIL TASK) ..... 10
FIG 11 PATTERN.V (PASS TASK) ..... 11
FIG 12 TESTBED.V(TESTBENCH) ..... 12
FIG 13 10 PAIRS OF INPUTS ..... 13
FIG 14 NWAVE RESULT ..... 13
FIG 15 AREA REPORT ..... 14
FIG 16 SLACK REPORT ..... 15
FIG 17 POWER DISSIPATION ..... 16
FIG 18 AREA REPORT(AFTER SCAN CHAIN INSERT) ..... 18
FIG 19 SLACK REPORT(AFTER SCAN CHAIN INSERT) ..... 19
FIG 20 POWER DISSIPATION(AFTER SCAN CHAIN INSERT) ..... 20
FIG 21 FAULT COVERAGE ..... 22

分工:

蕭方凱:RTL、Testbench、Compiler、DFT、Report

尤弘瑋: Algorithm、RTL、Testbench、Compiler、ATPG

(a) Write the RTL code in Verilog or VHDL that takes in two 8-bit positive integers, A[7:0] and B[7:0], and produces its quotient Q[7:0] and remainder R[7:0].

**divider.v**

```
1   module divider(dividend, divisor, quotient, remainder, clk, rst_n, in_valid, out_valid);
2   input clk, rst_n;
3   input [7:0] dividend;
4   input [7:0] divisor;
5   input in_valid;
6   output reg out_valid;
7   output reg [7:0] quotient;
8   output reg [7:0] remainder;
9
10  reg [7:0] input_A;
11  reg [7:0] input_B;
12  reg [2:0] c_state, n_state;
13  parameter IDLE = 2'b00;
14  parameter CALCULATE = 2'b01;
15  parameter OUTPUT = 2'b10;

17  always @(posedge clk or negedge rst_n) begin
18      if(!rst_n) begin
19          input_A <= 0;
20          input_B <= 0;
21      end
22      else begin
23          if(in_valid) begin
24              input_A <= dividend;
25              input_B <= divisor;
26          end
27          else if(c_state == CALCULATE) begin
28              input_A <= input_A - input_B;
29          end
30      end
31  end
32
33  always@(posedge clk or negedge rst_n)begin
34      if(!rst_n) c_state<=IDLE;
35      else c_state<=n_state;
36  end
```

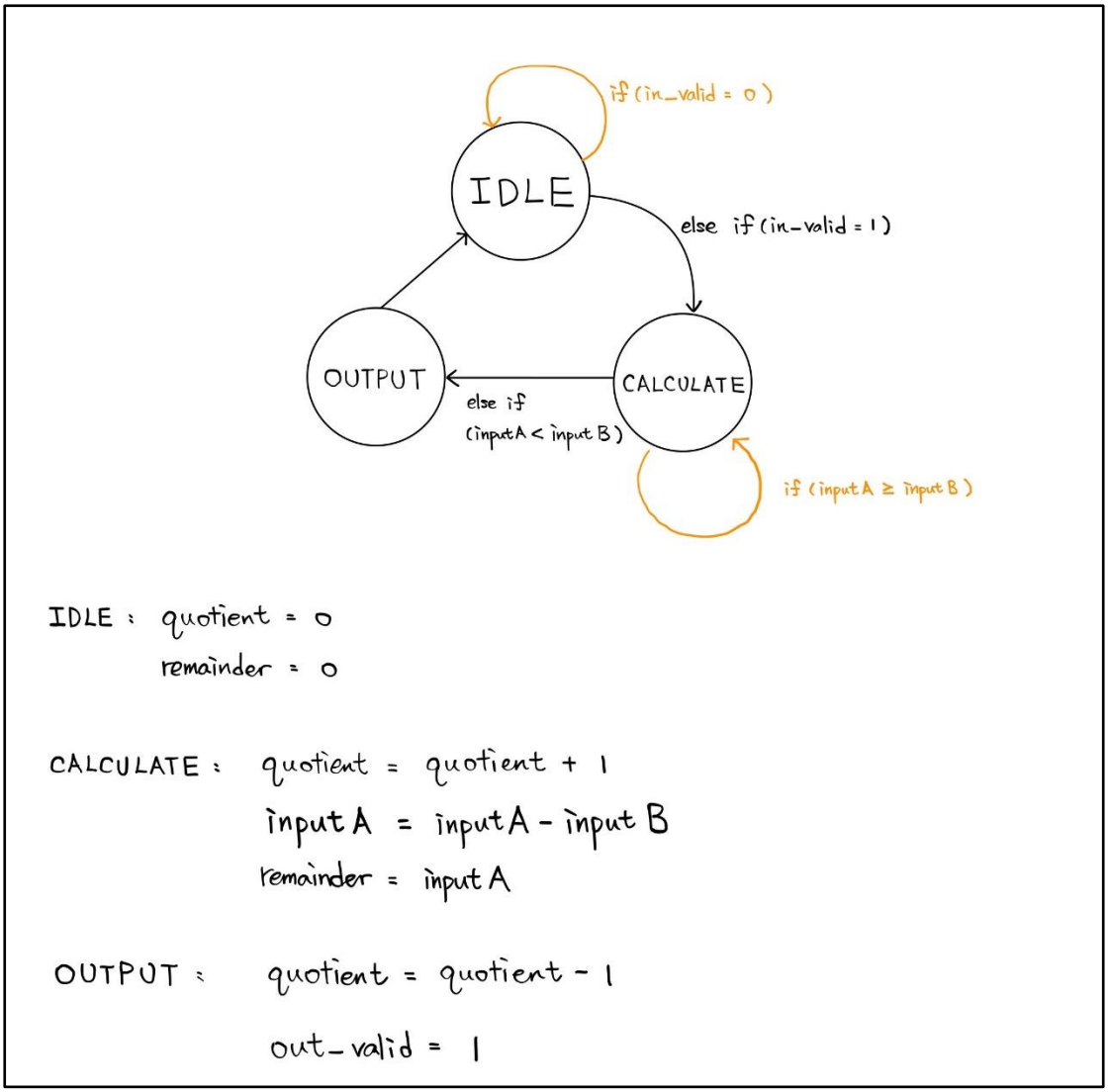
Fig 1 RTL code(divider.v)

```
38  always @(*) begin
39
40      case(c_state)
41
42          IDLE: begin
43
44              if(in_valid) n_state = CALCULATE;
45              else n_state = IDLE;
46          end
47
48          CALCULATE: begin
49
50              if(input_A >= input_B) n_state = CALCULATE;
51              else n_state = OUTPUT;
52          end
53
54          OUTPUT: begin
55
56              n_state = IDLE;
57          end
58
59          default: n_state = c_state;
60      endcase
61  end

68  always @(posedge clk or negedge rst_n) begin
69      if(!rst_n) quotient<=0;
70      else begin
71          if(c_state==IDLE) quotient<=0;
72          else if (c_state == CALCULATE) quotient <= quotient + 1'b1;
73          else if(c_state==OUTPUT) quotient <= quotient - 1;
74          else quotient<=quotient;
75      end
76  end
77
78  always @(posedge clk or negedge rst_n) begin
79      if(!rst_n) remainder<=0;
80      else begin
81          if(c_state==IDLE) remainder<=0;
82          else if(c_state==CALCULATE) remainder <= input_A;
83          else remainder<=remainder;
84      end
85  end
86
87
88  always @(posedge clk or negedge rst_n)begin
89      if(!rst_n) out_valid<=0;
90      else begin
91          if(c_state == OUTPUT) out_valid<=1;
92          else out_valid <= 0;
93      end
94
95  end
96
97  endmodule
```

Fig 2 RTL code(divider.v)

## Finite State Machine




另外，除了 FSM 功能以外，RTL 電路還包含以下主要功能:

1. rst_n = 0 時:

   input_A = 0; input_B = 0; c_state = IDLE; n_state = IDLE; quotient = 0;

   remainder = 0; out_valid = 0

2. in_valid = 1 時:

   將由 pattern 產生的 dividend 及 divisor 輸入給 input_A 及 input_B。

(b) Verify the correctness of your RTL code by a testbench. You should try it out by at least 3 pairs of input numbers.

**PATTERN.v**

```
1   `timescale 1ns/1ps
2   `define CYCLE_TIME 1.6
3
4   module PATTERN(
5       //OUTPUT signals
6       dividend,
7       divisor,
8       clk,
9       rst_n,
10      in_valid,
11      //INPUT signals
12      quotient,
13      remainder,
14      out_valid
15  );
16  //======================================
17  //          I/O PORTS
18  //======================================
19  output reg            clk;
20  output reg          rst_n;
21  output reg [7:0] dividend;
22  output reg [7:0]  divisor;
23  output reg       in_valid;
24  input           out_valid;
25  input [7:0]      quotient;
26  input [7:0]     remainder;
27
28  //======================================
29  //      PARAMETERS & VARIABLES
30  //======================================
31  parameter CYCLE      = `CYCLE_TIME;
32  parameter PATNUM     = 10;          PATNUM: number of inputs = 10
33  integer   SEED       = 122;
34  // PATTERN CONTROL
35  integer           i;
36  integer     err_cnt;
37  integer out_latency;
38
39
40  reg [7:0]  tp_quotient;
41  reg [7:0] tp_remainder;
```

Fig 3 PATTERN.v (I/O ports and parameters)

```
43  //=====================================
44  //              Clock
45  //=====================================
46  initial clk = 1'b0;
47  always #(CYCLE/2.0) clk = ~clk;
48
49  //=====================================
50  //              MAIN
51  //=====================================
52  initial main_task;
53
54  //=====================================
55  //              TASKS
56  //=====================================
57  task main_task; begin
58      reset_task;
59      $display("\n\t\t\t\t  DIVIDEND    DIVISOR   |  DIVIDEND   DIVISOR  |   QUOTIENT  REMAINDER  |  QUOTIENT  REMAINDER ");
60      $display("\t\t\t\t  --------   ---------  |  --------  ---------  |   --------  ---------  |  --------  ---------");
61      for(i = 0; i < PATNUM; i = i+1) begin
62          input_task;
63          check_task;
64      end
65      if (err_cnt !== 0) fail_task;
66      else pass_task;
67  end endtask
```

Fig 4 PATTERN.v (main task structure)

Fig 5 PATTERN.v (clock generation)

此 pattern 分別有 reset_task、input_task、check_task、error_task、fail_task、pass_task，及一些 display 以讓使用者觀察出輸入與輸出的對應結果。

以下為各個 task 之功能說明:

- Reset_task: 負責將所有訊號歸零，歸零後始 rst_n 常態保持在 1。
- Input_task: 負責 input 被除數及除數，並算出正確的商及餘數。
- Check_task: 驗證 pattern 算出的商及餘數是否與 RTL 算出的相同，若相同代表 RTL 正確，否則 RTL 存在錯誤。
- Error_task: display 錯誤的 input 發生在第幾筆 input，同時使 err_cnt 加 1，此 err_cnt 代表錯誤筆數。
- Fail_task: 顯示總共有幾筆錯誤資料，同時顯示出失敗小熊圖案。
- Pass_task: 顯示 PASS!! No errors were found!，同時顯示出成功小熊圖案。

```
69  //************************************
70  //      Reset Task
71  //************************************
72  task reset_task; begin
73      clk   = 0;
74      rst_n       = 1;
75      in_valid    = 0;
76      err_cnt     = 0;
77      #(CYCLE/2.0) rst_n = 0;
78      #(CYCLE/2.0) rst_n = 1;
79  end endtask
80
81  //************************************
82  //      Input Task
83  //************************************
84  task input_task; begin
85      in_valid = 1;   //in_valid 為1時才輸入
86      dividend = {$random(SEED)} % 255;
87      divisor = {$random(SEED)} % 255 + 1; // +1防止除數為0
88      tp_quotient = dividend / divisor;
89      tp_remainder = dividend % divisor;
90      $write("\tInput%d\t  %b    %b        %d       %d        ", i, dividend, divisor, dividend, divisor);
91      repeat(1) @(negedge clk);
92      in_valid = 0;
93      dividend = 8'bx;
94      divisor = 8'bx;
95  end endtask
```

Fig 6 PATTERN.v (reset task)

Fig 7 PATTERN.v (input task)

```
97  //************************************
98  //      Check Task
99  //************************************
100 task check_task; begin
101     out_latency = 0;
102     while(out_valid == 0)begin
103         out_latency = out_latency + 1;
104         @(negedge clk);   //只要out_valid 還沒拉起來就要多等一個cycle
105     end
106     if(out_valid)begin
107         $display("%b   %b       %d        %d       \n",  quotient, remainder,  quotient, remainder);
108         if ((tp_quotient !== quotient) || (tp_remainder !== remainder)) error;
109     end
110     repeat(5) @(negedge clk);
111 end endtask
112
113 //************************************
114 //      Error Task
115 //************************************
116 task error; begin
117     $display("\t\t\t ----------------------------------------------------------------------------------------");
118     $display("\t\t\t ERROR AT %d,  correct quotient should be %d\t\t   your quotient is %d", i, tp_quotient, quotient);
119     $display("\t\t\t ----------------------------------------------------------------------------------------\n");
120     err_cnt = err_cnt + 1;
121 end endtask
```

Fig 8 PATTERN.v (check task)

Fig 9 PATTERN.v (error task)

```
//************************************
//      Failed Task
//************************************
task fail_task; begin
    $display("\nFAIL!! There were %d errors in all.\n", err_cnt);
    $display("                                                                    ");
    $display("                                                 ./+oo+/.           ");
    $display("                                               /s:-----+s`          ");
    $display("                                               y/-------:y          ");
    $display("                                           `.-:/od+/------y`        ");
    $display("                                  `:///+++ooooooo+//::::-----:/y+:`  ");
    $display("                                -m+:::::::----------------::o+.     ");
    $display("                               `hod--------------------------:o+    ");
    $display("                          ./++/:s/-o/-------------------------/s///::.");
    $display("                        /s::-://--:-------------------------:oo/::::o+");
    $display("                      -+ho++++//hh:-----------------------------:s:-------+/");
    $display("                   -s+shdh+::+hm+-------------------------------+/--------:s");
    $display("                 -s:hMMMMNy---+y/-------------------------------:---------//");
    $display("                y:/NMMMMMN:---:s-/o:-------------------------------------+`");
    $display("                h--sdmmdy/-------:hyssoo++:---------------------------:/`");
    $display("                h---::::----------+oo+/::/+o:---------------------:+++s-`");
    $display("                s:----------------/s+///-----------------------------o`");
    $display("          ``..../s-----------------::-------------------------------o");
    $display("       -/oyhyyyyyym:----------------://////:--------------------------:/");
    $display("      /dyssyyyssssyh:-------------/o+/::::/+o/-----------------------+`");
    $display("    -+o/---:/oyyssshd/-----------+o:--------:oo---------------------:/.");
    $display("  `++--------:/syssddy+:-------/+------------s/-----------------:://`");
    $display(" .s:---------:+ooyysyyddoo++os-:s-------------/y----------------:++.");
    $display(" s:------------/yyhssyshy:---/:o:-------------:dsoo++//:::::-::+syh`");
    $display("`h--------------shysssyyms+oyo:--------------/hyyyyyyyyyyysyhyyyy`");
    $display("`h--------------:yysssyyhhyy+----------------+dyyyysssssssyyyhs+/.");
    $display(" s:--------------/yysssssyhy:-----------------shyyyyyhyyssssyyh.");
    $display(" .s---------------+sooosyyo------------------/yssssssyyyyssssyo");
    $display("  /+-------------------:++------------------:ysssssssssssssss-");
    $display("  `s+---------------------------------------:syssssssssssssyo");
    $display("`+yhdo--------------------:/--------------:syssssssssssssyy.");
    $display("+yysyhh:-------------------+o------------/ysyssssssssssssy/");
    $display(" /hhysyds:------------------y-----------/+yyssssssssssssyh`");
    $display(" .h-+yysyds:---------------:s----------:--/yssssssssssssym:");
    $display(" y/---oyyyyhyo:-----------:o:-------------:ysssssssyyyyssyyd-");
    $display("`h------+syyyyhhsoo+///+osh---------------:ysssyysyyyyyssssyd:");
    $display("/s--------:+syyyyyyyyyyyyyyhso/:-------::+oyyyyhyyyyyssssssyy+-");
    $display("+s-----------:/osyyyssssssyyyyhyyyyyyyydhyyyyyyssssssssyys/`");
    $display("+s---------------:/osyyyyssssssssssssssyyhyyssssssyyyyso/y`");
    $display("/s--------------------:/+ossyyyyyyssssssssyyyyyyyyso+:----:+");
    $display(".h-------------------------:::/++ooooooo+++/:::----------o`");
    repeat(5) @(negedge clk);
    $finish;
end endtask
```

Fig 10 PATTERN.v (fail task)

```
//**************************************
//      PASS Task
//**************************************
task pass_task; begin
    //$display("\nPASS!! No errors were found!\n");
    $display("\033[1;33m                `oo+oy+`                    \033[1;35m Congratulation!!! \033[1;0m                                ");
    $display("\033[1;33m              /h/----+y        `+++++:       \033[1;35m PASS This Lab........Maybe \033[1;0m                       ");
    $display("\033[1;33m            .y------:m/+ydoo+:y:---:+o      \033[1;35m Total Latency : %-10d\033[1;0m                             ", out_latency);
    $display("\033[1;33m             o+------/y--::::::+oso+:/y                                                                          ");
    $display("\033[1;33m             s/-----:/:----------:+ooy+-                                                                         ");
    $display("\033[1;33m           /o---------------/yhyo/::/o+/:-.`                                                                     ");
    $display("\033[1;33m          `ys----------------:::--------:::+yyo+                                                                 ");
    $display("\033[1;33m          .d/:-------------------:--------/--/hos/                                                               ");
    $display("\033[1;33m          y/-------------------::ds------:s:/-:sy-                                                               ");
    $display("\033[1;33m        +y--------------------::os:-----:ssm/o+`                                                                 ");
    $display("\033[1;33m       `d:-----------------------:-----/+o++yNNmms                                                               ");
    $display("\033[1;33m       /y----------------------------------hMMMMN.                                                              ");
    $display("\033[1;33m       o+---------------------:://:----------:odmdy/+.                                                           ");
    $display("\033[1;33m       o+---------------------::y:-----------::+o-/h                                                             ");
    $display("\033[1;33m       :y------------------------+s:------------/h:-:d                                                           ");
    $display("\033[1;33m       `m/-----------------------+y/----------:oy:--/y                                                           ");
    $display("\033[1;33m        /h------------------------:os++/:::/+o/:--:h-                                                            ");
    $display("\033[1;33m      `:+ym-------------------------://++++o/:---:h/                                                            ");
    $display("\033[1;31m     `hhhhhoooo++oo+/:\033[1;33m---------------------:oo----\033[1;31m+dd+                                     ");
    $display("\033[1;31m     shyyyhhhhhhhhhhso/:\033[1;33m---------------:+/---\033[1;31m/ydyyhs:`                                    ");
    $display("\033[1;31m     .mhyyyyyyhhhhdddhhhhhs+:\033[1;33m----------------\033[1;31m:sdmhyyyyyyo:                                 ");
    $display("\033[1;31m     `hhdhhyyyyhhhhhddddhyyyyyo++/:\033[1;33m--------\033[1;31m:odmyhmhhyyyyhy                                ");
    $display("\033[1;31m     -dyyhhyyyyyyhdhyhhddhhyyyyyhhhs+/::\033[1;33m-\033[1;31m:ohdmhdhhhdmdhdmy:                               ");
    $display("\033[1;31m      hhdhyyyyyyyyydyyyyhddhhhyyyyyhhhyyhdhdyyhyys+ossyhssy:-`                                               ");
    $display("\033[1;31m      `Ndyyyyyyyyyymdyyyyyyyhddddhhhyhhhhhhhhy+/:\033[1;33m------::/+o++++-`                                 ");
    $display("\033[1;31m       dyyyyyyyyyyyhNyydyyyyyyyyyyhhhhyyhhy+/\033[1;33m-----------------:/ooo:`                               ");
    $display("\033[1;31m      :myyyyyyyyyyyyNyhmhhhyyyyyhdhyyyhho/\033[1;33m-------------------------:+o/`                            ");
    $display("\033[1;31m     /dyyyyyyyyyyyyyddmmhyyyyyyhhyyyhh+:\033[1;33m---------------------------:+s-                             ");
    $display("\033[1;31m   +dyyyyyyyyyyyyyyydmyyyyyyyyyyyyyds:\033[1;33m-------------------------------:s+                             ");
    $display("\033[1;31m   -ddhhyyyyyyyyyyyyydyyyyyyyyyyyhd+\033[1;33m-----------------------------------:oo             `-++o+:.`       ");
    $display("\033[1;31m    `/dhshdhyyyyyyyyyhdyyyyyyyyyydh:\033[1;33m---------------------------------s/           -o/:/:/+s          ");
    $display("\033[1;31m     os-:/oyhhhhyyyydhyyyyyyyyyds:\033[1;33m-------------------------------------:h:--.`     `y:------+os        ");
    $display("\033[1;33m     h+-----\033[1;31m:/+oosshdyyyyyyyyhds\033[1;33m-----------------------------------+h//o+s+-.` :o------s/y      ");
    $display("\033[1;33m     m:-----------\033[1;31mdyyyyyyyyymo\033[1;33m-----------------------------------oh----://++oo------:s/d      ");
    $display("\033[1;33m     `N/----------+\033[1;31mmyyyyyyyyyo\033[1;33m-----------------------------------sy---------:/s------+o/d      ");
    $display("\033[1;33m    .m----------:d\033[1;31mhhyyyyyyd+\033[1;33m-------------------------------------y+----------+:-----oo/h      ");
    $display("\033[1;33m   +s----------+N\033[1;31mhmyyyyhd/\033[1;33m---------------------------------------:h:----------::-----+o/m      ");
    $display("\033[1;33m   h/----------:d/\033[1;31mmmhyyh:\033[1;33m-----------------------------------------oo-----------------+o/h      ");
    $display("\033[1;33m  `y----------so /\033[1;31mNhydh:\033[1;33m-------------------------------------------/h:-------------------:soo      ");
    $display("\033[1;33m `.:+o:--------+h   \033[1;31mmddhhh/:\033[1;33m---------------:/osssssoo+/::---------------+d+//++///::+++//::::::/y+`  ");
    $display("\033[1;33m -s+/::/--------+d.   \033[1;31mohso+/+y/:\033[1;33m----------:yo+/:-----:/oooo/:----------:+s//::-.....-::://////+/:`    ");
    $display("\033[1;33m s/-----------/y`          `/oo:--------:y/-------------:/oo+:------:/s:                                       ");
    $display("\033[1;33m o+:--------::++`            `:so/:-----s+----------------:oy+:--:+s/``````                                    ");
    $display("\033[1;33m  :+o++//+oo/.                  .+o+::--os------------------:oy+oo:`/o++++o-                                  ");
    $display("\033[1;33m     .---.`                       -+oo/:yo:--------------------:oy-:h/:---:+oyo                                ");
    $display("\033[1;33m                                    `:+omy/----------------------+h:-----:y+//so                               ");
    $display("\033[1;33m                                       `-ys:--------------------+s-----+s///om                               ");
    $display("\033[1;33m                                          -os+::---------------/y-----ho///om                               ");
    $display("\033[1;33m                                             -+oo//:-----------:h-----h+///+d                               ");
    $display("\033[1;33m                                                `-oyy+:---------s:----s/////y                               ");
    $display("\033[1;33m                                                   `-/o+::-----:+----oo///+s                               ");
    $display("\033[1;33m                                                      ./+o+::--------:y///s:                               ");
    $display("\033[1;33m                                                         ./+oo/-----oo/+h                                  ");
    $display("\033[1;33m                                                            `://++++syo`                                   ");
    $display("\033[1;0m");
    repeat(5) @(negedge clk);
        $finish;
end endtask

endmodule
```

Fig 11 PATTERN.v (pass task)

**TESTBED.v**

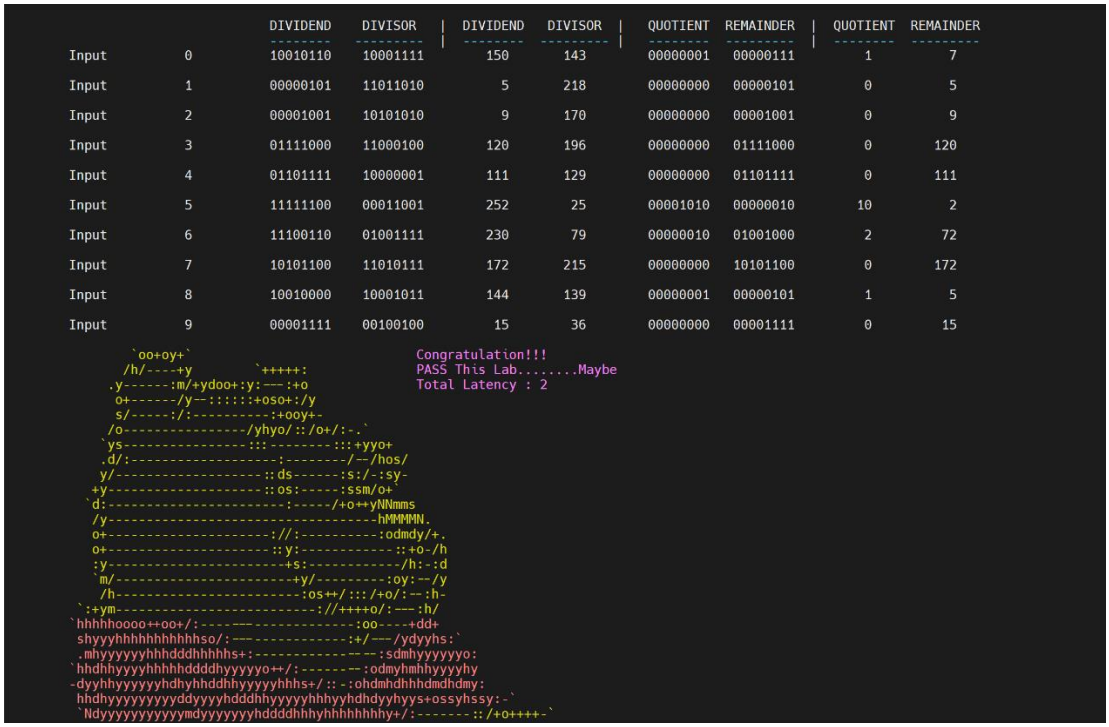
```
1   `include "PATTERN.v"
2   module TESTBED;
3
4   wire clk;
5   wire rst_n;
6   wire in_valid;
7   wire [7:0] dividend;
8   wire [7:0] divisor;
9   wire [7:0] quotient;
10  wire [7:0] remainder;
11  wire out_valid;
12
13  divider U_divider(
14      .dividend(dividend),
15      .divisor(divisor),
16      .quotient(quotient),
17      .remainder(remainder),
18      .clk(clk),
19      .rst_n(rst_n),
20      .in_valid(in_valid),
21      .out_valid(out_valid)
22  );
23
24  PATTERN U_PATTERN(
25      .dividend(dividend),
26      .divisor(divisor),
27      .quotient(quotient),
28      .remainder(remainder),
29      .clk(clk),
30      .rst_n(rst_n),
31      .in_valid(in_valid),
32      .out_valid(out_valid)
33  );
34
35  initial begin
36      $sdf_annotate("divider_syn.sdf", U_divider);
37      $fsdbDumpfile("./divider.fsdb");
38      $fsdbDumpvars(0,"+mda");
39      $fsdbDumpvars();
40  end
41  endmodule
```

RTL PART

PATTERN PART

Fig 12 TESTBED.v(testbench)

## Display on MobaXterm



| | | DIVIDEND | DIVISOR | DIVIDEND | DIVISOR | QUOTIENT | REMAINDER | QUOTIENT | REMAINDER |
|---|---|---|---|---|---|---|---|---|---|
| Input | 0 | 10010110 | 10001111 | 150 | 143 | 00000001 | 00000111 | 1 | 7 |
| Input | 1 | 00000101 | 11011010 | 5 | 218 | 00000000 | 00000101 | 0 | 5 |
| Input | 2 | 00001001 | 10101010 | 9 | 170 | 00000000 | 00001001 | 0 | 9 |
| Input | 3 | 01111000 | 11000100 | 120 | 196 | 00000000 | 01111000 | 0 | 120 |
| Input | 4 | 01101111 | 10000001 | 111 | 129 | 00000000 | 01101111 | 0 | 111 |
| Input | 5 | 11111100 | 00011001 | 252 | 25 | 00001010 | 00000010 | 10 | 2 |
| Input | 6 | 11100110 | 01001111 | 230 | 79 | 00000010 | 01001000 | 2 | 72 |
| Input | 7 | 10101100 | 11010111 | 172 | 215 | 00000000 | 10101100 | 0 | 172 |
| Input | 8 | 10010000 | 10001011 | 144 | 139 | 00000001 | 00000101 | 1 | 5 |
| Input | 9 | 00001111 | 00100100 | 15 | 36 | 00000000 | 00001111 | 0 | 15 |

Congratulation!!!
PASS This Lab.....Maybe
Total Latency : 2

Fig 13 10 pairs of inputs

由左至右，分別為:

由 pattern 所產生的被除數及除數(binary) → 由 pattern 所產生的被除數及除數(decimal) → RTL 的商數及餘數(binary) → RTL 的商數及餘數(decimal)

## nWave

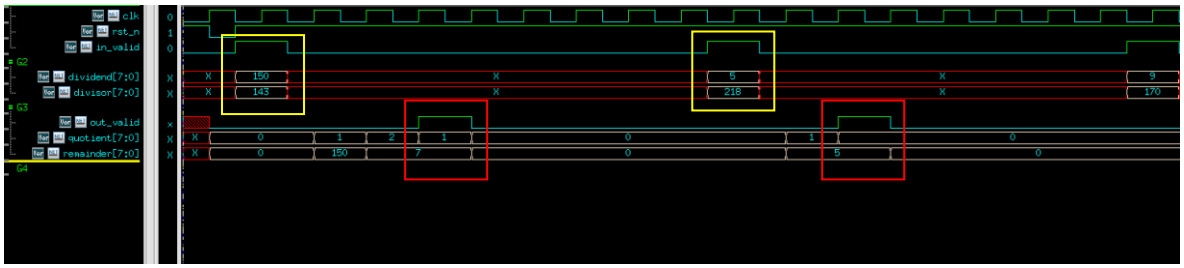



Fig 14 nWave result

一開始先將 rst_n 變為 0，使電路 reset。Reset 完後，將 rst_n 拉回 1，同時將 in_valid enable，使 pattern 提供一組被除數及除數，待 RTL 運算完成後，out_valid 變為 1，此時的 quotient 及 remainder 即為輸出結果值。

以上面波形為例，第一筆輸入為黃色部分 150/143，得到的結果為 1…7; 第二筆輸入為 5/218，得到的結果為 0…5。

並且從波形上可看出，latency = 2 cycles。

(c) Use a synthesis script to convert your RTL code into a gate-level netlist. Report the final gate count, the maximum operating speed (in MHz) and the estimated power dissipation in (mW) using Design Compiler.

**Area Report**

```
Number of ports:                          78
Number of nets:                          324
Number of cells:                         267
Number of combinational cells:           229
Number of sequential cells:               35
Number of macros/black boxes:              0
Number of buf/inv:                        61
Number of references:                     61

Combinational area:               932.097619
Buf/Inv area:                     137.592005
Noncombinational area:            642.096017
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 1574.193636
Total area:                 undefined
1
```

Fig 15 area report

**Total Area = 1574.194**

$$\textbf{Final Gate Count} = \frac{\boldsymbol{Total\ area}}{\boldsymbol{area\ of\ NAND2\ gate}} = \frac{\mathbf{1574.19}}{\mathbf{2.8}} = \mathbf{562}$$

## Slack Report

```
Point                                       Incr       Path
-------------------------------------------------------------
clock clk (rise edge)                       0.00       0.00
clock network delay (ideal)                 0.00       0.00
input_B_reg_0_/CK (DFFRHQX2)                0.00       0.00 r
input_B_reg_0_/Q (DFFRHQX2)                 0.15       0.15 r
sub_28/B[0] (divider_DW01_sub_1)            0.00       0.15 r
sub_28/U104/Y (INVX2)                       0.04       0.19 f
sub_28/U91/Y (NOR2X4)                       0.07       0.26 r
sub_28/U98/Y (OAI21X2)                      0.06       0.31 f
sub_28/U111/Y (CLKNAND2X2)                  0.03       0.34 r
sub_28/U112/Y (NAND2XL)                     0.06       0.40 f
sub_28/U126/Y (XNOR2X1)                     0.08       0.48 f
sub_28/DIFF[3] (divider_DW01_sub_1)         0.00       0.48 f
U98/Y (AO2B2BX2)                            0.13       0.61 f
U116/Y (AO21X4)                             0.10       0.71 f
input_A_reg_3_/D (DFFRHQX4)                 0.00       0.71 f
data arrival time                                      0.71

clock clk (rise edge)                       0.80       0.80
clock network delay (ideal)                 0.00       0.80
input_A_reg_3_/CK (DFFRHQX4)                0.00       0.80 r
library setup time                         -0.09       0.71
data required time                                     0.71
-------------------------------------------------------------
data required time                                     0.71
data arrival time                                     -0.71
-------------------------------------------------------------
slack (MET)                                            0.00
```

Fig 16 Slack report

## Clock period = 0.8ns

**Maximum Operating speed** $= \frac{1000}{0.8+0} MHz = 1250MHz$

## Power dissipation

```
Global Operating Voltage = 0.9
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  = 743.7397 uW   (96%)
  Net Switching Power  =  31.4377 uW    (4%)
                         ---------
Total Dynamic Power    = 775.1775 uW  (100%)

Cell Leakage Power     =   6.1710 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000           0.0000  (   0.00%)
memory             0.0000           0.0000              0.0000           0.0000  (   0.00%)
black_box          0.0000           0.0000              0.0000           0.0000  (   0.00%)
clock_network      0.0000           0.0000              0.0000           0.0000  (   0.00%)
register           0.7041        6.7734e-03          1.9327e+06           0.7128  (  91.23%)
sequential         0.0000           0.0000              0.0000           0.0000  (   0.00%)
combinational   3.9659e-02        2.4664e-02          4.2383e+06        6.8561e-02  (   8.77%)
--------------------------------------------------------------------------------------------------
Total              0.7437 mW     3.1438e-02 mW     6.1710e+06 pW         0.7813 mW
1
```

Fig 17 power dissipation

**Total Power Dissipation = 0.7813mW**

(d) Add the scan chain into your gate-level netlist obtained by part(c), report the resulting gate count, the maximum operating speed (in MHz) of your circuit. Compare to the non-scan version, and report the area overhead percentage and performance penalty due to scan chain insertion.

```
---------------------------------------------------------------------
  DRC Report

  Total violations: 0

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


Test Design rule checking did not find violations

---------------------------------------------------------------------
  Sequential Cell Report

  0 out of 35 sequential cells have violations

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

SEQUENTIAL CELLS WITHOUT VIOLATIONS
      *  35 cells are valid scan cells
```

- **DRC Report: Total violations: 0**
- **Sequential Cell Report: no violation**

## Area Report

```
Number of ports:                           80
Number of nets:                           326
Number of cells:                          267
Number of combinational cells:            229
Number of sequential cells:                35
Number of macros/black boxes:               0
Number of buf/inv:                         61
Number of references:                      62

Combinational area:               985.723218
Buf/Inv area:                     147.470405
Noncombinational area:            881.294407
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 1867.017625
Total area:                 undefined
1
```

Fig 18 area report(after scan chain insert)

**Total Area = 1867.018**

**Final Gate Count** $= \frac{\boldsymbol{Total\ area}}{\boldsymbol{area\ of\ NAND2\ gate}} = \frac{\mathbf{1867.02}}{\mathbf{2.8}} = \mathbf{667}$

**Overhead Percentage** $= \frac{\mathbf{1867.02}}{\mathbf{1574.19}} = \mathbf{1.186}$

## Slack Report

```
Point                                    Incr       Path
-------------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.00       0.00
input_A_reg_3_/CK (SDFFRHQX4)            0.00       0.00 r
input_A_reg_3_/Q (SDFFRHQX4)             0.13       0.13 f
U4/Y (NAND2BX8)                          0.08       0.20 f
U1/Y (INVX6)                             0.03       0.23 r
U24/Y (OAI32X1)                          0.07       0.30 f
U115/Y (OAI2B2X2)                        0.09       0.39 r
U17/Y (OAI22X2)                          0.07       0.46 f
U83/Y (AND2X4)                           0.07       0.54 f
U2/Y (NOR2X2)                            0.06       0.60 r
U13/Y (AND2X6)                           0.07       0.67 r
U12/Y (MXI2X6)                           0.03       0.70 f
c_state_reg_0_/D (SDFFRHQX8)             0.00       0.70 f
data arrival time                                   0.70

clock clk (rise edge)                    0.80       0.80
clock network delay (ideal)              0.00       0.80
c_state_reg_0_/CK (SDFFRHQX8)            0.00       0.80 r
library setup time                      -0.11       0.69
data required time                                  0.69
-------------------------------------------------------------
data required time                                  0.69
data arrival time                                  -0.70
-------------------------------------------------------------
slack (VIOLATED)                                   -0.02
```

Fig 19 Slack report(after scan chain insert)

## Clock period = 0.8ns

**Maximum Operating speed** $= \frac{1000}{0.8 + 0.02} MHz = 1219.512 MHz$

## Power dissipation

```
Global Operating Voltage = 0.9
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =   1.0117 mW   (97%)
  Net Switching Power  =  36.6594 uW    (3%)
                         ---------
Total Dynamic Power    =   1.0483 mW  (100%)

Cell Leakage Power     =   7.9324 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000           0.0000  (   0.00%)
memory             0.0000           0.0000              0.0000           0.0000  (   0.00%)
black_box          0.0000           0.0000              0.0000           0.0000  (   0.00%)
clock_network      0.0000           0.0000              0.0000           0.0000  (   0.00%)
register           0.9701        9.8548e-03          3.0478e+06           0.9830  (  93.07%)
sequential         0.0000           0.0000              0.0000           0.0000  (   0.00%)
combinational  4.1538e-02        2.6805e-02          4.8846e+06       7.3227e-02  (   6.93%)
--------------------------------------------------------------------------------------------------
Total              1.0117 mW     3.6659e-02 mW     7.9324e+06 pW          1.0563 mW
1
```

Fig 20 power dissipation(after scan chain insert)

**Total Power Dissipation = 1.0563mW**

## Comparing before/after scan chain

| | Before scan chain | After scan chain | Overhead Percentage |
|---|---|---|---|
| **Area** | 1574.194 | 1867.018 | 15.684% |
| **Operating speed** | 1250MHz | 1219.512MHz | 2.5% |
| **Power dissipation** | 0.7813mW | 1.0563mW | 26.03% |

## Comment:

本組在 clk period 上試了很多組合，因 operation speed 及 area 是一個 trade off，無法同時兼顧兩者。

以下為幾組不同結果(均為 before insert scan chain):

1. 在 clk period=2ns 時(預設)，divider 面積約為 1167，power 為 0.202mW。
2. 在 clk period=1.5ns 時，divider 面積約為 1205，power 為 0.271mW。
3. 在 clk period=1ns 時，divider 面積約為 1394，power 為 0.518mW。
4. 在 clk period=0.8ns 時，divider 面積約為 1574，power 為 0.781mW。

**由上面幾組數據可得到在在 clk period=2ns 時，有最好的整體表現，雖然頻率不快，但面積及功耗都算小，是個不錯的設計。**
**但因此作業不會比較數據，且在合成 RTL 電路時盡可能使 slack 為 0 會比較好，故本作業還是使用 clk period = 0.8ns 為報告數據。**

(e) Run ATPG using a commercial tool available and report the fault coverage.

**Fault Coverage:**

```
    Uncollapsed Stuck Fault Summary Report
-----------------------------------------------
fault class                     code   #faults
------------------------------  ----  ---------
Detected                         DT       1952
Possibly detected                PT          0
Undetectable                     UD          6
ATPG untestable                  AU          0
Not detected                     ND          0
-----------------------------------------------
total faults                              1958
test coverage                           100.00%
-----------------------------------------------
```

Fig 21 Fault Coverage