時序電路設計及應用

Homework 1

姓名：王品然

學號：113063572

Generate a linear function y=1000-30*x, where x is the value of an 4-bit digital code x[3:0]. As a result, y is a value in the range of [550, 1000]. Try to develop a circuit that can find a proper value of x so that the value of y is closest to a given target value, *target*. You are advised to use 10 bits for variable y, i.e., y[9:0].

(a) Develop a software program in any software language (e.g., C/C++ or Python) that can solve the above problem, using the **successive approximation scheme** discussed in class. Verify your program by trying target values of 630 and 780. Report the proper values of x[7:0] for each case.

[71]:
```python
#y_t stands for target y
y_t = 630

def y(x):
    return 1000 - 30 * x

initial_x = 0b1000
x = initial_x
i = 0
x_pin = 0

while i < 3:
    if y(x) > y_t:
        x_pin = x
        x = x | x >> 1
    else:
        x = (x_pin + x) // 2
    i = i + 1

# 輸出結果
print(f'approximation y: {y(x)}')
print(f'approximation x: {x}')
```

```
approximation y: 610
approximation x: 13
```

[73]:
```python
#y_t stands for target y
y_t = 780

def y(x):
    return 1000 - 30 * x

initial_x = 0b1000
x = initial_x
i = 0
x_pin = 0

while i < 3:
    if y(x) > y_t:
        x_pin = x
        x = x | x >> 1
    else:
        x = (x_pin + x) // 2
    i = i + 1

# 輸出結果
print(f'approximation y: {y(x)}')
print(f'approximation x: {x}')
```

```
approximation y: 790
approximation x: 7
```

(b) Convert your software subroutine into a synthesizable RTL code (in Verilog or VHDL). Verify your RTL code with a testbench. Make sure that the results are consistent with those produced by your software program.

RTL code:

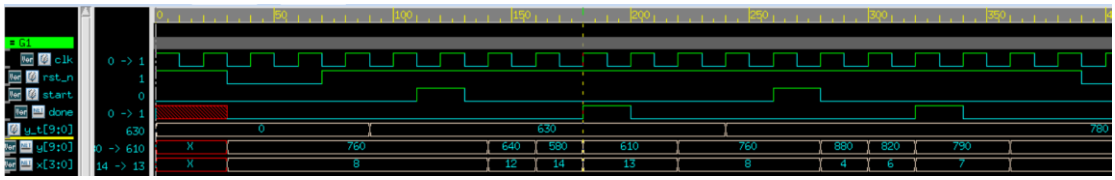
```
module sa (
    output [9:0] y,
    output reg [3:0] x,
    output       done,
    input        clk,
    input        rst_n,
    input  [9:0] y_t,
    input        start
);

//////////////////////
//DECLARATION
//////////////////////
//state
localparam ST_IDLE = 2'b00;
localparam ST_SA = 2'b01;
localparam ST_DONE = 2'b10;

reg   [1:0] state;
reg   [1:0] state_nx;

//control signals
wire        more_less;

//count index
reg   [1:0] cnt;        //發生三次SA運算
wire  [1:0] cnt_next;
wire        cnt_en;
wire        cnt_clear;
wire        cnt_next_full;

//////////////////////
//FSM
//////////////////////
//setting next state
always@(posedge clk or negedge rst_n)begin
   if(~rst_n)  state <= ST_IDLE;
   else        state <= state_nx;
end


//decision of next state
always@(*)begin
   state_nx = state;
   case(state)
   ST_IDLE   : if(start)       state_nx = ST_SA;
   ST_SA     : if(cnt == 2'b01)  state_nx = ST_DONE;
   ST_DONE   : state_nx = ST_IDLE;
   default: state_nx = state;
   endcase
end

//output controlled by FSM
assign done = (state == ST_DONE);

//counter
assign cnt_en        = (state == ST_SA);
assign cnt_clear     = (state == ST_DONE);
assign cnt_next      = cnt_en? cnt-2'b1: cnt;

always@(posedge clk or negedge rst_n)begin
   if(~rst_n)     cnt <= 2'b11;
   else if(cnt_clear) cnt <= 2'b11;
   else           cnt <= cnt_next;
end

//////////////////////
//Successive Approximation
//////////////////////
//x value decision
always@(posedge clk or negedge rst_n) begin
    if(~rst_n)    x <= 4'b1000;
    else begin
        case(state)
            ST_IDLE  : x <= 4'b1000;
            ST_SA    :
                    begin
                        x[cnt_next] <= 1;
                        if (~more_less) x[cnt] <= 0;
                        else x[cnt] <= x[cnt];
                    end
            ST_DONE  : x <= x;
            default  : x <= x;
        endcase
    end
end

//more_less
assign more_less = (y > y_t)? 1 : 0;

//y assignment
assign y = 10'd1000 - (5'd30 * x);


endmodule
```

Note: 儘管運算的方式跟 python code 略有不同，但結果會是一樣的。在 python 時我是使用相加除以 2 的方式逼近，而在 RTL code 中我是使用，在逼近時改變其中兩個 bit 以實現同樣的功能，會與 python 不同的原因是因為這樣的寫法可以更加簡潔，並預期有更小的面積與更小的 propagation delay。

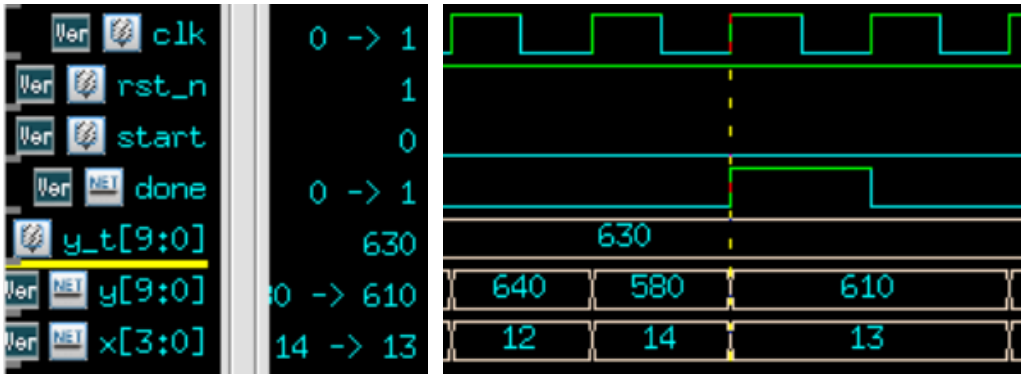
Testbench:

```
`timescale 1ns / 100ps
module testbench;

    localparam period = 2;
    localparam delay = 1;

    wire [9:0] y;
    wire [3:0] x;
    wire       done;
    reg        clk;
    reg        rst_n;
    reg  [9:0] y_t;
    reg        start;

    sa sa (
        .y(y),
        .x(x),
        .done(done),
        .clk(clk),
        .rst_n(rst_n),
        .y_t(y_t),
        .start(start)
    );

    initial begin
        $fsdbDumpfile("../4.Simulation_Result/sa_rtl.fsdb");
        $fsdbDumpvars;
    end

    always #(period / 2) clk = ~clk;

    initial begin
        clk = 1;
        rst_n = 1;
        start = 0;
        y_t = 0;
        #(period + delay) rst_n = 0;
        #(period * 2) rst_n = 1;
        #(period) y_t = 630;
        #(period) start = 1;
        #(period) start = 0;
        @(negedge done);

        @(posedge clk);
        #(period) y_t = 780;
        #(period) start = 1;
        #(period) start = 0;
        @(negedge done);

        @(posedge clk);
        #(delay * 3) rst_n = 0;
        #(period * 8) $finish;
    end

    // Automatically finish
    initial begin
        #200;
        $finish;
    end

endmodule
```

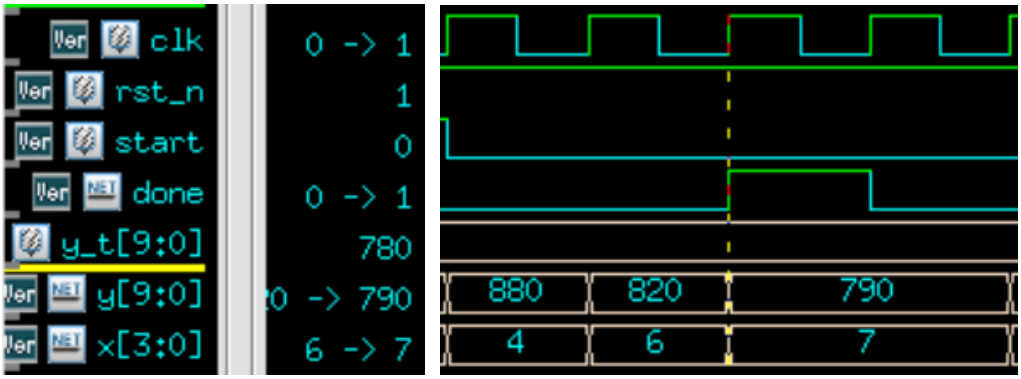
Waveform (RTL simulation):



當 target (也就是我定義的 y_t)為 630 時，done 拉起時，x 有正確輸出為 13，與 python 結果相同，y 值也是與之相符的 610。此段結果放大圖如下：



當 target (y_t)為 780 時，done 拉起時，x 有正確輸出為 7，與 python 結果相同，y 值也是與之相符的 790。此段結果放大圖如下：

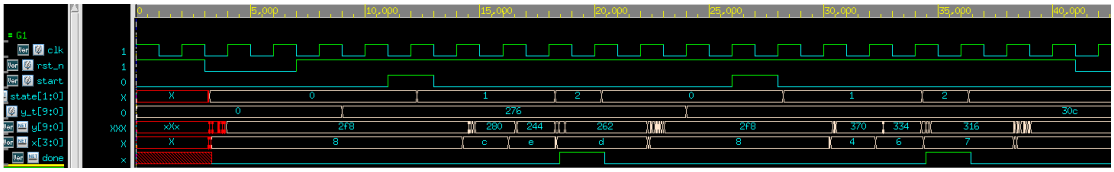


(c) Use a synthesis script to convert your RTL code into a gate-level netlist. Perform gate-level simulation to verify the gate-level netlist again, to ensure that its behavior is identical to your previous RTL code.

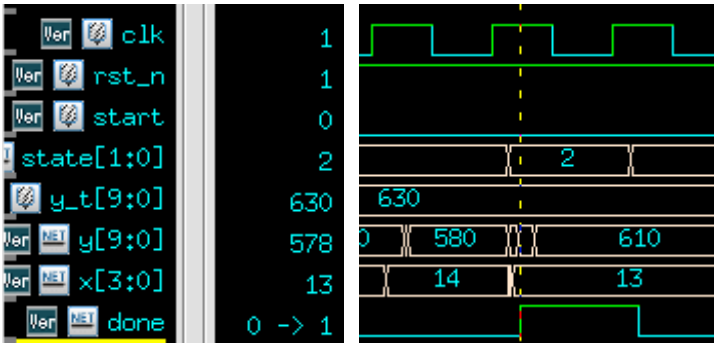
使用範例中提供的.tcl 檔，只改其中檔案的路徑以及 clock cycle，輸出 gate-level netlist 如下：

```
module sa ( y, x, done, clk, rst_n, y_t, start );
  output [9:0] y;
  output [3:0] x;
  input [9:0] y_t;
  input clk, rst_n, start;
  output done;
  wire   N33, N34, N46, N47, n19, n22, n30, n32, n330, n340, n35, n36, n37,
         n42, n43, n44, n45, n460, n470, n49, n50, n51, n52, n53, n54, n55,
         n56, n57, n58, n59, n60, n61, n62, n63, n64, n65, n66, n67, n68, n69,
         n70, n71, n72, n73, n74, n75, n76, n77, n78, n79, n80, n81, n82, n83,
         n84, n85, n86, n87, n88, n89, n90, n91, n92, n93, n94, n95, n96, n97,
         n98, n99, n100, n101, n102, n103, n104, n105, n106, n107, n108, n110,
         n111;
  wire   [1:0] state;

  DFFRQX2 state_reg_0_ ( .D(n36), .CK(clk), .RN(rst_n), .Q(state[0]) );
  DFFRQX2 x_reg_2_ ( .D(n330), .CK(clk), .RN(rst_n), .Q(x[2]) );
  DFFRQX2 x_reg_1_ ( .D(n340), .CK(clk), .RN(rst_n), .Q(x[1]) );
  DFFSQXL cnt_reg_0_ ( .D(N46), .CK(clk), .SN(rst_n), .Q(N33) );
  DFFSQXL cnt_reg_1_ ( .D(N47), .CK(clk), .SN(rst_n), .Q(N34) );
  DFFSQXL x_reg_3_ ( .D(n32), .CK(clk), .SN(rst_n), .Q(x[3]) );
  DFFRQX2 x_reg_0_ ( .D(n35), .CK(clk), .RN(rst_n), .Q(y[1]) );
  DFFRX1 state_reg_1_ ( .D(n37), .CK(clk), .RN(rst_n), .Q(state[1]), .QN(n110)
        );
  INVX2 U38 ( .A(1'b1), .Y(y[0]) );
  INVX2 U40 ( .A(1'b0), .Y(y[9]) );
  OR3XL U42 ( .A(n72), .B(n43), .C(n71), .Y(n42) );
  CLKNAND2X2 U43 ( .A(n42), .B(n74), .Y(y[5]) );
  XNOR2X1 U44 ( .A(x[3]), .B(n68), .Y(n71) );
  INVX2 U45 ( .A(n56), .Y(n68) );
  AOI21BXL U46 ( .A0(n68), .A1(y[8]), .B0N(n107), .Y(n43) );
  INVX4 U47 ( .A(y[1]), .Y(n107) );
  OAI2BB1X1 U48 ( .A0N(n98), .A1N(state[0]), .B0(n110), .Y(n99) );
  OAI32XL U49 ( .A0(n82), .A1(n81), .A2(n87), .B0(y[7]), .B1(n80), .Y(n92) );
  OAI2BB1XL U50 ( .A0N(n68), .A1N(y[8]), .B0(n107), .Y(n73) );
  OAI2BB1X1 U51 ( .A0N(n58), .A1N(n107), .B0(n59), .Y(n69) );
  MXI4XL U52 ( .A(y[1]), .B(x[2]), .C(x[1]), .D(x[3]), .S0(N34), .S1(N33), .Y(
        n95) );
  AOI32XL U53 ( .A0(y_t[8]), .A1(x[3]), .A2(y_t[9]), .B0(y_t[9]), .B1(n93),
        .Y(n94) );
  OA22X2 U54 ( .A0(n92), .A1(n91), .B0(y_t[8]), .B1(x[3]), .Y(n93) );
  OAI221XL U55 ( .A0(y_t[2]), .A1(n62), .B0(y_t[1]), .B1(n107), .C0(n65), .Y(
        n63) );
  CLKINVX2 U56 ( .A(n55), .Y(n57) );
  CLKINVX2 U57 ( .A(n78), .Y(n77) );
  NAND2BX1 U58 ( .AN(x[1]), .B(n43), .Y(n75) );
  AOI21BX1 U59 ( .A0(x[1]), .A1(n73), .B0N(n75), .Y(n45) );
  XNOR2XL U60 ( .A(x[1]), .B(y[1]), .Y(n58) );
  NAND2BXL U61 ( .AN(N33), .B(n105), .Y(n106) );
  AO2B2X2 U62 ( .B0(n22), .B1(x[2]), .A0(n106), .A1N(n104), .Y(n330) );
  OAI222XL U63 ( .A0(N33), .A1(n104), .B0(n103), .B1(n105), .C0(n102), .C1(
        n101), .Y(n340) );
  INVXL U64 ( .A(x[1]), .Y(n101) );
  XNOR2XL U65 ( .A(y[1]), .B(n60), .Y(n62) );
  OR3X1 U66 ( .A(n60), .B(y[1]), .C(n59), .Y(n44) );
  CLKNAND2X2 U67 ( .A(n44), .B(n69), .Y(y[3]) );
  INVXL U68 ( .A(n58), .Y(n60) );
  NAND2BXL U69 ( .AN(y_t[3]), .B(y[3]), .Y(n65) );
  OAI2BB1X1 U70 ( .A0N(n70), .A1N(n69), .B0(n43), .Y(n74) );
  INVX2 U71 ( .A(n69), .Y(n72) );
  INVX2 U72 ( .A(y[5]), .Y(n84) );
  XNOR2X1 U73 ( .A(n70), .B(n72), .Y(y[4]) );
  OR2X2 U74 ( .A(n45), .B(n74), .Y(n78) );
  OAI2BB1X1 U75 ( .A0N(y[6]), .A1N(n89), .B0(n83), .Y(n87) );
  OR2X2 U76 ( .A(n96), .B(n105), .Y(n97) );
  INVX2 U77 ( .A(n71), .Y(n70) );
  OAI2BB1X1 U78 ( .A0N(n45), .A1N(n74), .B0(n78), .Y(y[6]) );
  INVX2 U79 ( .A(y[3]), .Y(n61) );
  INVX2 U80 ( .A(n62), .Y(y[2]) );
  INVX2 U81 ( .A(n54), .Y(done) );
  INVX2 U82 ( .A(y_t[7]), .Y(n80) );
  INVX2 U83 ( .A(n85), .Y(n81) );
  OAI2BB1X1 U84 ( .A0N(y[4]), .A1N(n67), .B0(n66), .Y(n82) );
  NAND2BX2 U85 ( .AN(n95), .B(n94), .Y(n105) );
  NAND2BX2 U86 ( .AN(y_t[7]), .B(y[7]), .Y(n83) );
  XNOR2X1 U87 ( .A(x[2]), .B(n76), .Y(n79) );
  INVX2 U88 ( .A(n75), .Y(n76) );
  NAND2BX2 U89 ( .AN(x[2]), .B(n57), .Y(n56) );
  OAI2BB1X1 U90 ( .A0N(n100), .A1N(x[3]), .B0(n99), .Y(n32) );
  NAND3BX2 U91 ( .AN(n96), .B(N33), .C(n110), .Y(n100) );
  MX2X1 U92 ( .A(N34), .B(n97), .S0(N33), .Y(n98) );
  NAND2BX2 U93 ( .AN(x[1]), .B(n107), .Y(n55) );
  OAI32XL U94 ( .A0(n90), .A1(y[6]), .A2(n89), .B0(n88), .B1(n87), .Y(n91) );
  AOI32XL U95 ( .A0(y_t[4]), .A1(n86), .A2(n85), .B0(y_t[5]), .B1(n84), .Y(n88) );
  INVX2 U96 ( .A(n83), .Y(n90) );
  INVX2 U97 ( .A(y[4]), .Y(n86) );
  XNOR2X1 U98 ( .A(x[2]), .B(n57), .Y(n59) );
  OAI21X1 U99 ( .A0(N34), .A1(n111), .B0(n110), .Y(n22) );
  NAND2BX2 U100 ( .AN(y_t[5]), .B(y[5]), .Y(n85) );
  OAI32XL U101 ( .A0(n108), .A1(N34), .A2(n19), .B0(n470), .B1(n107), .Y(n35)
        );
  INVX2 U102 ( .A(n106), .Y(n108) );
  OAI2BB1X1 U103 ( .A0N(n65), .A1N(n64), .B0(n63), .Y(n66) );
  AO2B2X2 U104 ( .B0(y_t[3]), .B1(n61), .A0(y_t[2]), .A1N(y[2]), .Y(n64) );
  OA21X1 U105 ( .A0(N33), .A1(n111), .B0(n470), .Y(n102) );
  INVX2 U106 ( .A(x[3]), .Y(y[8]) );
  INVX2 U107 ( .A(N34), .Y(n96) );
  NAND2BX2 U108 ( .AN(state[1]), .B(state[0]), .Y(n19) );
  NAND2BX2 U109 ( .AN(n19), .B(N34), .Y(n104) );
  MX2X1 U110 ( .A(n52), .B(state[0]), .S0(n53), .Y(n36) );
  NOR2X2 U111 ( .A(state[1]), .B(n50), .Y(n52) );
  INVX2 U112 ( .A(n103), .Y(n50) );
  INVX2 U113 ( .A(n51), .Y(n53) );
  OAI2B11X2 U114 ( .A1N(start), .A0(state[1]), .B0(n54), .C0(n19), .Y(n51) );
  NAND2BX2 U115 ( .AN(state[0]), .B(state[1]), .Y(n54) );
  NAND2BX2 U116 ( .AN(n49), .B(n54), .Y(N47) );
  XNOR2X1 U117 ( .A(n30), .B(n96), .Y(n49) );
  NOR2X2 U118 ( .A(N33), .B(n19), .Y(n30) );
  NAND3BX2 U119 ( .AN(n111), .B(N33), .C(n96), .Y(n103) );
  INVX2 U120 ( .A(state[0]), .Y(n111) );
  OAI2BB1X1 U121 ( .A0N(n53), .A1N(state[1]), .B0(n103), .Y(n37) );
  NAND2X2 U122 ( .A(n460), .B(n54), .Y(N46) );
  XOR2X1 U123 ( .A(N33), .B(n19), .Y(n460) );
  AOI21BX2 U124 ( .A0(N34), .A1(state[0]), .B0N(n110), .Y(n470) );
  INVX2 U125 ( .A(y_t[6]), .Y(n89) );
  INVX2 U126 ( .A(y_t[4]), .Y(n67) );
  BUFX2 U127 ( .A(y[1]), .Y(x[0]) );
  AO2B2X4 U128 ( .B0(n79), .B1(n78), .A0(n77), .A1N(n79), .Y(y[7]) );
endmodule
```

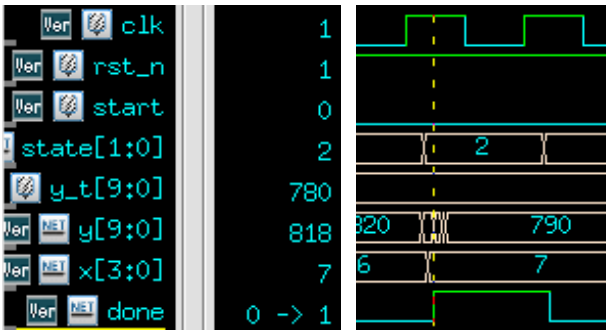
Testbench 架構與 synthesis 前的相同，只有多了 sdf 的 annotate，模擬的波型圖如下。Waveform (gate level simulation):



當 target (y_t)為 630 時，done 拉起時，x 有正確輸出為 13，與 python 結果相同，y 值也在短暫 delay 後，在同個 clock cycle 轉成與之相符的 610。此段結果放大圖如下：



當 target (y_t)為 780 時，done 拉起時，x 有正確輸出為 7，與 python 結果相同，y 值也在短暫 delay 後，在同個 clock cycle 轉成與之相符的 790。此段結果放大圖如下：



(d) Report the final gate count, the maximum operating speed (in MHz), and the estimated power consumption in (mW) of your design using *Design Compiler*.

1. 計算 gate count

從 synthesis 完成後的報告可以看到 total cell area 約為 599.76μm^2，如下表：

```
Combinational area:                452.995210
Buf/Inv area:                       72.676802
Noncombinational area:             146.764803
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                   599.760013
```

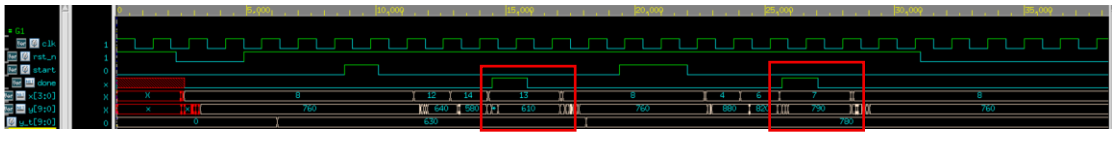
而從 tsmc090g.pdf 中查詢之 NAND2X1 area = $2.52 \times 1.12 = 2.8224(\mu m^2)$

Drive Strength	Height (um)	Width (um)
NAND2XL	2.52	1.12
NAND2X1	2.52	1.12

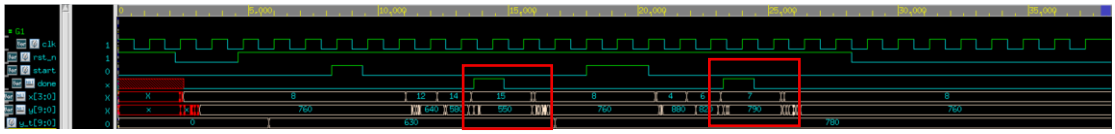
可以得到 **gate count** = $599.76 \div 2.8224 \approx \mathbf{212.5}$。

2. 尋找 max operating speed

藉由切換 testbench 中的 clock period 長度，並觀察模擬波型，去尋找什麼時候電路功能會出現錯誤。當 clock period 是 1.3ns 時，可以看到 waveform 結果依舊正確：



但當 clock period 是 1.2ns 時，可以看到 waveform 結果其中之一已經錯誤：



此時再檢查 clock period = 1.3ns 時會不會有 timing violation，如下表：

Point	Incr	Path
clock clk (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
x_reg_0_/CK (DFFRHQX2)	0.00	0.00 r
x_reg_0_/Q (DFFRHQX2)	0.14	0.14 f
U119/Y (NAND2BX4)	0.10	0.24 f
U99/Y (INVXL)	0.04	0.28 r
U92/Y (NAND2BX2)	0.05	0.33 f
U43/Y (XOR2X1)	0.08	0.41 r
U97/Y (OAI2BB1X4)	0.10	0.51 r
U80/Y (OR2X4)	0.08	0.58 r
U79/Y (OAI2BB1X2)	0.06	0.64 f
U56/Y (OAI2BB1XL)	0.16	0.80 f
U53/Y (OR3X2)	0.13	0.94 f
U54/Y (OR2X4)	0.08	1.01 f
U84/Y (OAI2B11X4)	0.04	1.05 r
U51/Y (NAND2BX2)	0.05	1.10 f
U48/Y (INVX1)	0.03	1.13 r
U148/Y (OAI32X1)	0.06	1.20 f
x_reg_0_/D (DFFRHQX2)	0.00	1.20 f
data arrival time		1.20
clock clk (rise edge)	1.30	1.30
clock network delay (ideal)	0.00	1.30
x_reg_0_/CK (DFFRHQX2)	0.00	1.30 r
library setup time	-0.10	1.20
data required time		1.20
data required time		1.20
data arrival time		-1.20
slack (MET)		0.00

發現仍沒有 violation，因此可以推測 max operating speed 約為 **769MHz**。

3. 查找 estimated power consumption

Design Compiler 給出的 estimated power consumption 如下表所示：

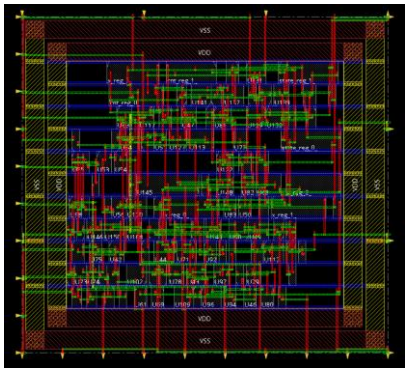
Power Group	Internal Power	Switching Power	Leakage Power	Total Power (%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000 (0.00%)	
memory	0.0000	0.0000	0.0000	0.0000 (0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000 (0.00%)	
clock_network	0.0000	0.0000	0.0000	0.0000 (0.00%)	
register	9.8965e-02	4.2075e-03	4.7681e+05	0.1036 (75.29%)	
sequential	0.0000	0.0000	0.0000	0.0000 (0.00%)	
combinational	1.9224e-02	1.2581e-02	2.2184e+06	3.4023e-02 (24.71%)	
Total	0.1182 mW	1.6788e-02 mW	2.6952e+06 pW	0.1377 mW	

1

綜合以上，gate count 約為 **212.5**，max operating speed 約為 **769MHz**，total power consumption 約為 **0.1377mW**。

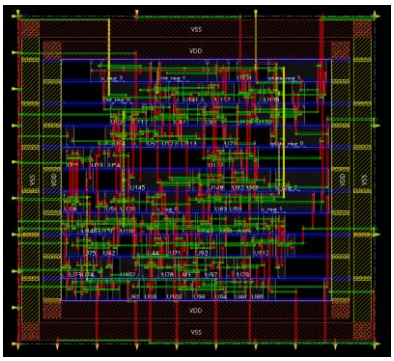
(e) Generate the layout by running some Automatic Placement and Routing Tool (e.g., SoC Encounter or the like). Report the size of the layout. Perform post-layout simulation or analysis and report again the maximum operating speed and power consumption. Compare how these results are different from those in pre-layout analysis.

Pre-CTS layout and setup time checking result after optimization:



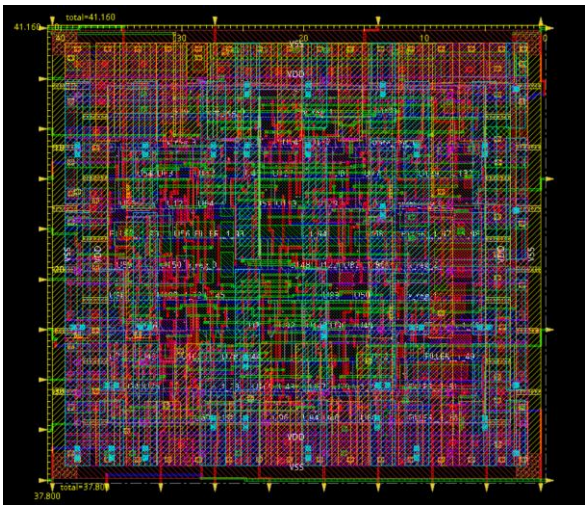
```
    optDesign Final Summary

Setup views included:
 func_mode_max

+--------------------+--------+--------+--------+
|     Setup mode     |  all   | reg2reg | default |
+--------------------+--------+--------+--------+
|           WNS (ns):|  3.111 |  3.111 |  4.000 |
|           TNS (ns):|  0.000 |  0.000 |  0.000 |
|    Violating Paths:|    0   |    0   |    0   |
|          All Paths:|   16   |    8   |   14   |
+--------------------+--------+--------+--------+
```

Post-CTS layout and setup/hold time checking result after optimization:



```
    optDesign Final Summary

Setup views included:
 func_mode_max
Hold  views included:
 func_mode_min

+--------------------+--------+--------+--------+
|     Setup mode     |  all   | reg2reg | default |
+--------------------+--------+--------+--------+
|           WNS (ns):|  3.110 |  3.110 |  4.029 |
|           TNS (ns):|  0.000 |  0.000 |  0.000 |
|    Violating Paths:|    0   |    0   |    0   |
|          All Paths:|   16   |    8   |   14   |
+--------------------+--------+--------+--------+

+--------------------+--------+--------+--------+
|     Hold mode      |  all   | reg2reg | default |
+--------------------+--------+--------+--------+
|           WNS (ns):|  0.003 |  0.131 |  0.003 |
|           TNS (ns):|  0.000 |  0.000 |  0.000 |
|    Violating Paths:|    0   |    0   |    0   |
|          All Paths:|   16   |    8   |   14   |
+--------------------+--------+--------+--------+
```

Layout after design for manufacturing:



由圖中尺標計算 layout 面積約為 1555.48μm^2

Verify geometry result:

```
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                        ..... bin size: 3840
  VERIFY GEOMETRY ..... SubArea : 1 of 1
  VERIFY GEOMETRY ..... Cells          :  0 Viols.
  VERIFY GEOMETRY ..... SameNet        :  0 Viols.
  VERIFY GEOMETRY ..... Wiring         :  0 Viols.
  VERIFY GEOMETRY ..... Antenna        :  0 Viols.
VG: elapsed time: 1.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.
```

Verify connectivity result:

```
******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)
```

Verify antenna result:

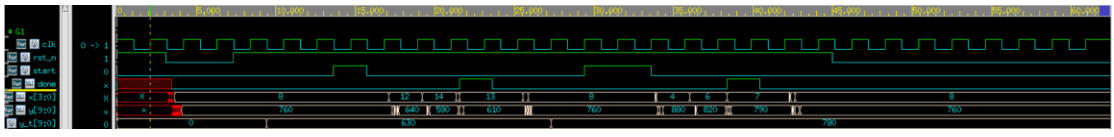
```
innovus 10> verifyProcessAntenna -reportfile gcd.antenna.rpt -error 1000

******* START VERIFY ANTENNA ********
Report File: gcd.antenna.rpt
LEF Macro File: sa.antenna.lef
Verification Complete: 0 Violations
******* DONE VERIFY ANTENNA ********
(CPU Time: 0:00:00.0  MEM: 0.000M)
```

Power analysis:

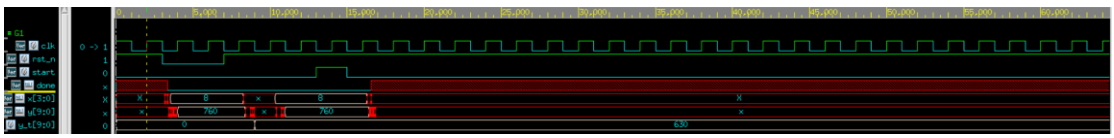
```
Total Power
-----------------------------------------------------------------
----------
Total Internal Power:         0.06635405            58.3552%
Total Switching Power:        0.04600630            40.4603%
Total Leakage Power:          0.00134685             1.1845%
Total Power:                  0.11370719
-----------------------------------------------------------------
```

Post-layout simulation waveform:

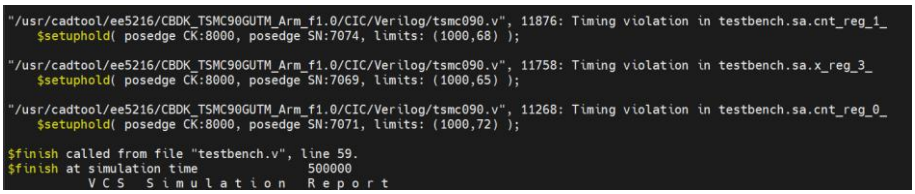
Clock period = 2.1ns



Clock period = 2ns



由 waveform 可以得知當 clock period 為 2.1ns 時，電路功能仍維持正確，且當 clock period 為 2ns 時，simulation 時出現 timing violation 的 message：



綜合以上推測 max operating speed 約為 476MHz、layout 面積約 1555.48μm^2、total power consumption 約為 0.1137mW。與 pre-layout simulation 的比較如下：

	pre-layout	post-layout
Max Operating Speed (MHz)	769	476
Area (μm^2)	599.76	1555.48
Total Power (mW)	0.1377	0.1137

可以發現操作頻率在 post-sim 時大幅下降，理由可能是因為在實際 layout 過後，計入更多的寄生電容、電阻，使得 propagation delay 上升。面積在 layout 後也大幅超出預期，可能是因為在合成時主要考量 cell 大小而沒有正確計入 routing, clock tree, power ring 等等效應，使得實際 layout 面積比合成時來得大不少。最後在估計 power consumption 方面，雖然數字沒有差多少，但細看 internal, switching power 的占比會發現兩者間的差異甚大，猜測可能因為在不同 tool 間估算 power 的方式不同造成它們之間的差異。