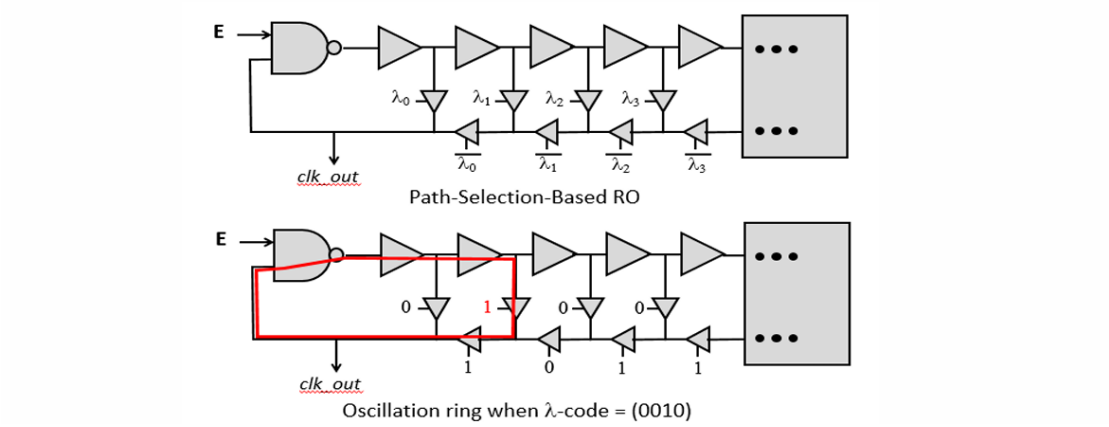
時序電路設計及應用

Homework 2

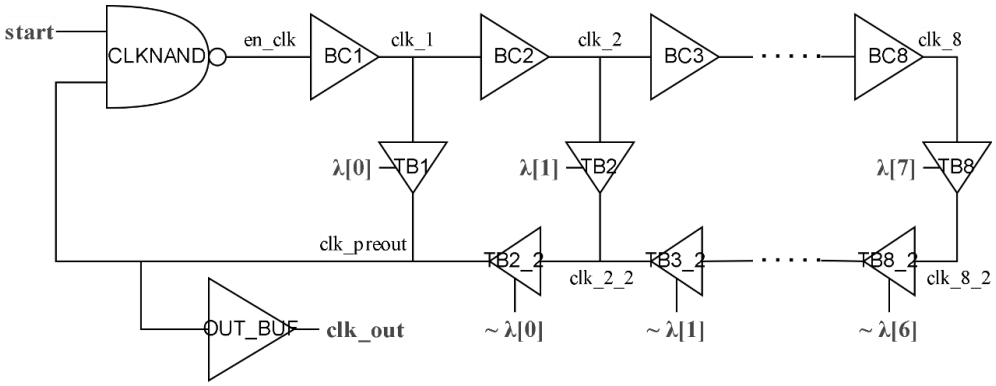
姓名：王品然

學號：113063572

(a) Design a **path-selection-based Digital Controlled Oscillator (DCO)** as indicated below. Report the oscillation frequency of your DCO under various control code values. Note that you may need to use a longer buffer chain and a control code with more bits than what is shown in the figure to ensure that the oscillation frequency fits your needs. (30%)

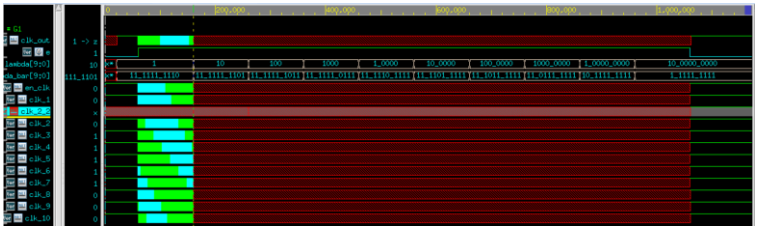


首先，在 APR 時，會因為 clk_out 有 multi-driver 的問題，導致合成 clock tree 遇到很多 error message，因此我將題目所指定的 pat-selection-based DCO 做了一點小改變，也就是在真正的 clk_out 前加一個 buffer 確保只有一個 driver 在驅動 clk_out，並且不太影響原本設計的 DCO 頻率，加完 buffer 後的 DCO 的結構如下圖：

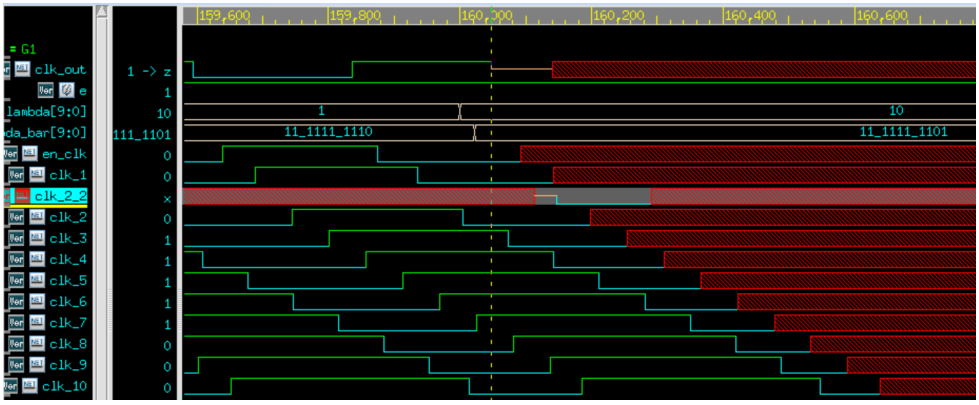


Note: 接下來的報告會運用上圖之節點名稱

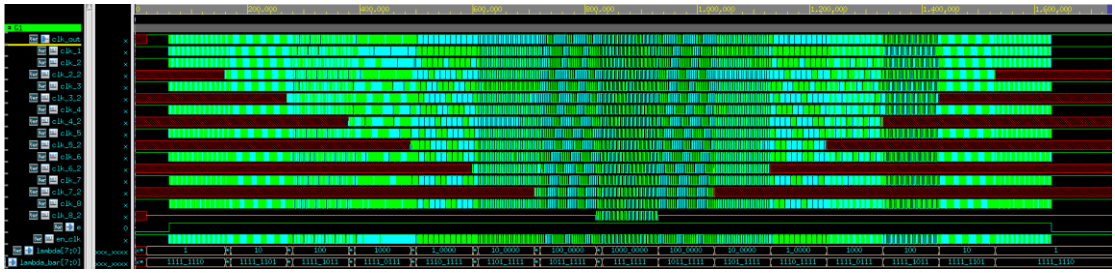
一開始我是使用 RTL code 寫出此架構並直接交由 DC 去合成出 netlist 和 sdf 後跑模擬，結果發現 waveform 不如預期(忘記截圖就把檔案覆蓋過去了)，儘管 clock 訊號有確實產生，但沒有辦法如預期中，隨著 λ-code 的變化而有遞增的 clock cycle，猜測原因是因為 DC 自動優化各路徑使得 path delay 沒有顯著差異，因此我選擇直接手刻 gate level 的 netlist 再交由 DC 生成出 sdf 檔後跑模擬，直接跳過 compile 的步驟讓 tool 不會動到任何 cell 或路徑，這次使用 1 顆 CLKBUFX2 作為 buffer cell，TBUFX2 作為 tristate buffer，NAND2X8 作為 NAND gate，INVX1 作為 λ-code 的 inverter，首次模擬的 waveform 如下：



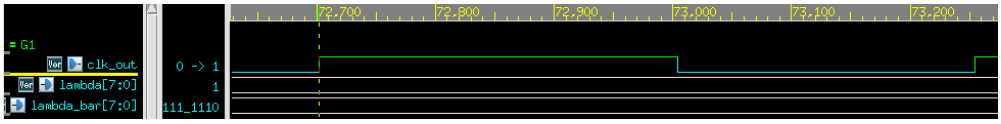
關鍵部分放大圖：



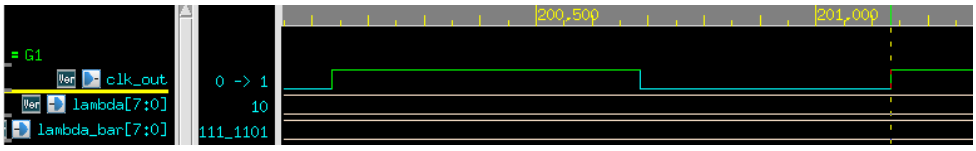
從 waveform 中或是 sdf 檔案中可以得知所產生的 clock cycle 過短，與目標的頻率有差距，且由上圖觀察到 NAND gate 的 delay 大約是 0.04ns 非常微小，tri-state buffer 的 delay 大約是 0.17ns。主要問題是在於 λ-code 做轉變時會造成訊號出現 unknown 的情形，推測是因為 tri-state buffer 的 delay 沒有短到可以忽略，使得 clk_2_2 節點尚未被 clk_2 驅動，在 λ-code 做變化時，傳遞了 unknown 的 clk_2_2 訊號出去，第二個可能原因是 λ-code 的 invert 訊號晚了一步到達，使得驅動同一節點的兩個 tri-state buffer 同時關閉(OE=0)，使得 output unknown。針對以上情況，我首先選擇使用更長的 buffer chain (2 個 X2 buffer) 作為 buffer chain delay cell，並且在 testbench 給入同時的 λ-code 的訊號以及 invert 訊號，接下來在 λ-code 調大時，需要預先開啟第一階段的 tri-state buffer，先給未知的節點做充放電，得到如下的 waveform：



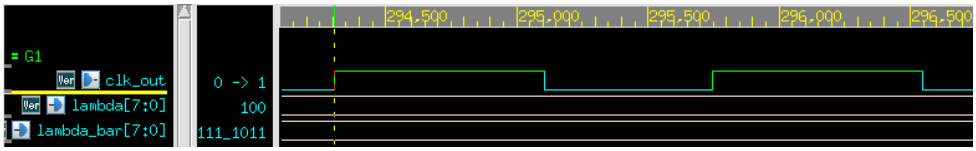
λ[7:0]=8’b00000001 時的局部放大圖，clock cycle = 0.521 (ns)



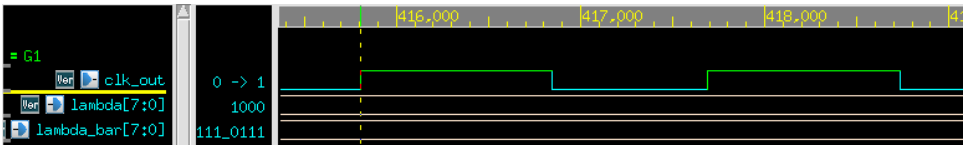
λ[7:0]=8’b00000010 時的局部放大圖，clock cycle = 0.961 (ns)



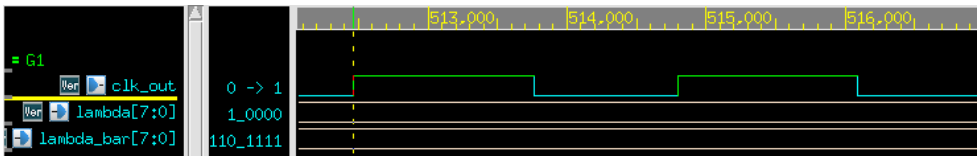
λ[7:0]=8’b00000100 時的局部放大圖，clock cycle = 1.401 (ns)



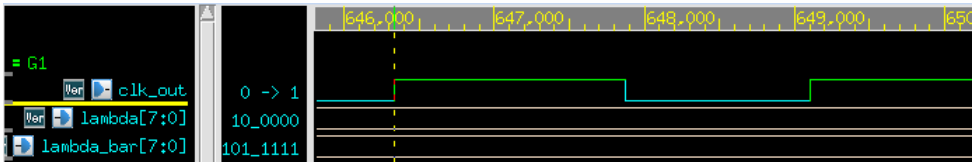
λ[7:0]=8’b00001000 時的局部放大圖，clock cycle = 1.841 (ns)



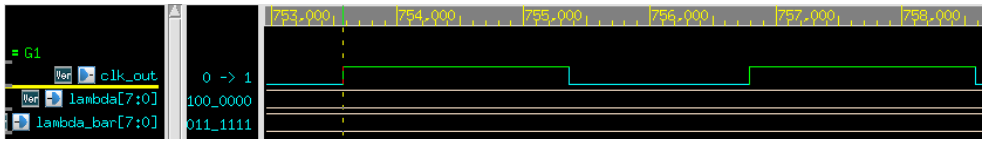
λ[7:0]=8’b00010000 時的局部放大圖，clock cycle = 2.281 (ns)



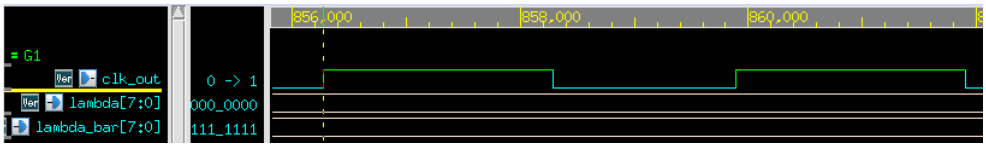
λ[7:0]=8’b00100000 時的局部放大圖，clock cycle = 2.721 (ns)



λ[7:0]=8’b01000000 時的局部放大圖，clock cycle = 3.161 (ns)



λ[7:0]=8’b10000000 時的局部放大圖，clock cycle = 3.566 (ns)



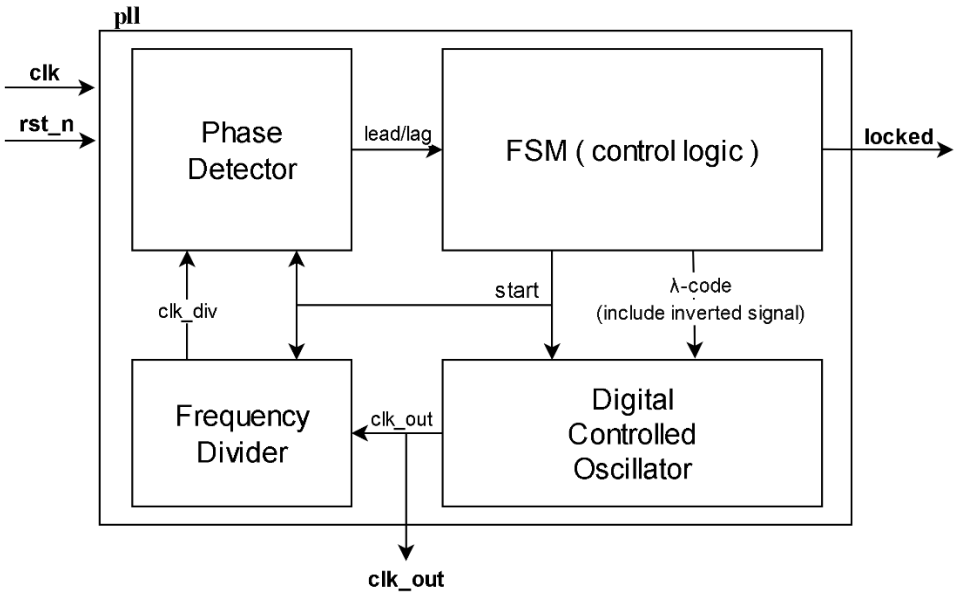
在 gate-sim 階段，沒有 jitter 的現象，不同 λ-code 所對應的頻率整理如下：

λ[7:0]	Clock Period (ns)	Clock Frequency (GHz)
0000_0001	0.521	1.92
0000_0010	**0.961**	**1.04**
0000_0100	1.401	0.71
0000_1000	1.841	0.54
0001_0000	2.281	0.44
0010_0000	2.721	0.37
0100_0000	3.161	0.32
1000_0000	3.566	0.28

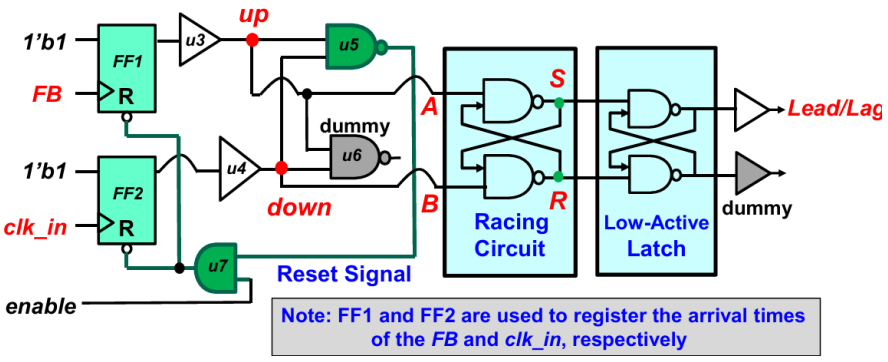
最終使用 buffer chain cell 是 2 個 CLKBUFX3；NAND gate 是 CLKNANDX2；tri-state buffer 是 TBUFX4；output buffer 是 CLKBUFX6。而使用 8 條 path 的 DCO 是因為 8 是 2 的冪次方，可以比較方便在第二題使用作業一的 successive approximation，且我們不需要太多條，因為題目只有要求 1GHz 的目標頻率而已，不用特地作出太大的 clock frequency range。

(b) Design a **frequency divider** and a **controller** that works with your **DCO** and some Phase Detector (**PD**) we have discussed in class so that your PLL will produce a frequency closest to the designated clock frequency after **frequency acquisition**. Verify your DLL design by Verilog simulation using as accurate delay model as possible for your DCO, frequency divider, and controller. Use the behavior model you have constructed in homework #1 for your PD. **Show the average clock cycle times of *clk_out* observed over 10 clock cycles after the frequency acquisition and its error as compared to the ideal clock cycle time at 1GHz (which is 1000ps).** (40%)

依據題意設計 Phased-Locked Loop (PLL)之 block diagram 如下圖所示：



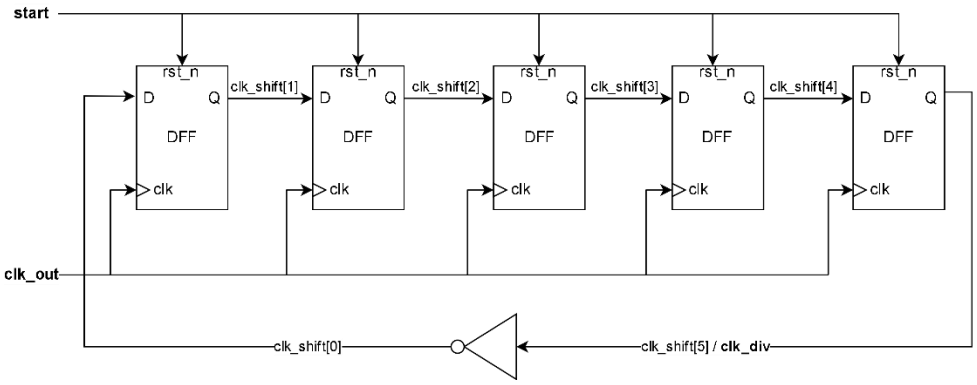
其中 Phase Detector (PD)使用上課提過的 based on the notion of racing 的電路來實現，主要原因是好奇這個電路能不能被正確的實現，因為其實這次的 DCO 能夠產生的頻率彼此相差不少，因此若使用單純一顆 DFF 當作 phase detector 應該也能成功，但這次常使使用如下圖之電路(截自講義)：



我會使用 start 訊號作為 enable

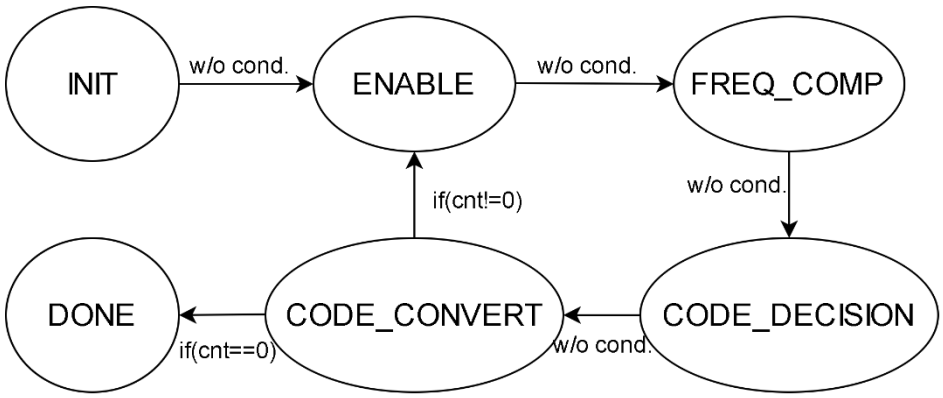
當實作這部分的電路時，我沒有選擇用直接手刻 gate-level netlist 並在後續的 flow 中採取 don't touch 的動作，因為我仍然希望工具可以優化這段路徑，再加上我沒有想要慢慢調整每顆 cell 的 size，增加實作的難度。

接下來是除頻器的部分，因為要除的頻率有可能是將近 2GHz 的高頻率，因此採用的電路必須是 combinational logic 非常簡單(短)的設計，考慮到這點，這次採用的是 5 顆 DFF 和一顆 inverter 組成的迴圈，並拉出最尾端的訊號作為除以 10 倍的頻率，結構如下圖所示：



將 start 訊號作為除頻器的 active-low async reset signal，當 start=0 時，clk_shift[5:1]被 reset 至 5'b11111，因此當 start 訊號為 1 時，clk_shift[5]會維持五個 clk_out 週期為 1，接著五個週期為 0，如此循環即可得到除以 10 倍頻的 clock 訊號，因此將 clk_shift[5]作為 clk_div 訊號送回給 PD 作判斷。這樣的除頻器好處是在於 combinational logic 幾乎等於沒有，不太可能造成 timing violation (hold time 的部分再給 tool 優化即可)。

最後是 control logic 的部分，我選擇使用 FSM 來完成此區塊的邏輯，以下是我設計的 FSM 與其解釋：



(1) INIT state: PLL reset 後會到這個 state，主要負責設定電路內的初始值。
執行: start<=0、x[2:0]=3'b100 和 cnt<=2。

(2) ENABLE state: 讓 DCO, PD, frequency divider 同時發動。
執行: start<=1。

(3) FREQ_COMP state: 維持 DCO, PD, frequency divider 同時運作的狀態，等待 lead/lag 訊號穩定確定下來。其實就像是一個緩衝而已，理論上不需要。
執行: start<=1。

(4) CODE_DECISION state: 依據 lead/lag 訊號，使用 successive approximation 決定 x[2:0]，並停下 DCO, PD, frequency divider。
執行: start<=0, x<=x_next 和 cnt<=cnt-1。

(5) CODE_CONVERT state: 將 x[2:0]轉換成 one-hot 的 λ[7:0]，並且如果 cnt 已經為 0，則進入 DONE state；若不為 0，則回到 ENABLE state 重新一輪的 loop。應該可以跟前面一個 state 直接合併，原本以為 clock cycle 會太短所以拆成 pipeline 進行，後來才想通這裡的 clock 是接 100MHz 的訊號，應該不用拆分那麼細也不會造成 setup time violation。
執行: λ<=λ_next。

(6) DONE state: 表示電路已經準備好目標頻率之 clock 訊號。
執行: start<=1, locked <=1。

我用的方法與我在作業一使用的方法幾乎一樣，也是靠著 counter 數 n 次(x 有 n 位就數 n 次)來確認 successive approximation 已經結束。另外，在 DCO 模擬時，雖然 λ-code 在往上增加時，應該需要預先打開 tri-state buffer 為未知節點充放電，但我在做 PLL 模擬時不管 λ 是上升或下降都沒有再遇到 unknown 的情況(可能是因為每次改變 λ 時，都讓 enable 訊號=0)，因此我沒有如同(a)小題一樣預先打開，但我還是有順便讓 λ 與其 inverter 訊號盡量同時到達，作法是讓他們最後都通過一級 DFF 再進入 DCO 中，但其實也可以說是 λ 與其 inverter 訊號在前一個 state 就已經準備好了。在 locked 之後 λ-code 就不再變動，因為這只是粗調的 stage，如果要做 phase tracking，clk_out 的變化幅度會過大。

在合成此電路時需要注意的一點是要 set_dont_touch 不希望被優化的 DCO cell，還有要注意的一點是有不同的 clock frequency 因此再請 DC 做 STA 時，要 create 不同的 clock，並且設成不同 group，我的作法如下：

```
create_clock -name CLK_REF -period 10 [get_ports clk]

create_clock -name CLK_OUT -period 0.5 [get_pins u_dco/clk_out]

create_generated_clock -name CLK_DIV \
    -source [get_pins u_dco/clk_out] \
    -divide_by 10 \
    [get_pins clk_shift[5]]

set_clock_groups -asynchronous \
    -group {CLK_REF} \
    -group {CLK_OUT CLK_DIV}
```

因為我設計的 DCO 最快可能跑到將近 2GHz，因此假設最極端的 clock cycle 下去做 STA。剩下的 tcl 檔內容與上次作業大同小異我就不放上來了。合成後的 report 如下：

(1) Area

```
Combinational area:                354.211210
Buf/Inv area:                       98.078403
Noncombinational area:             610.344007
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                   964.555217
```

(2) Timing

```
Startpoint: clk_shift_reg[1]
            (rising edge-triggered flip-flop clocked by CLK_OUT)
Endpoint: clk_shift_reg[2]
          (rising edge-triggered flip-flop clocked by CLK_OUT)
Path Group: CLK_OUT
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock CLK_OUT (rise edge)                0.00       0.00
clock network delay (ideal)              0.00       0.00
clk_shift_reg[1]/CK (DFFSQXL)            0.00       0.00 r
clk_shift_reg[1]/Q (DFFSQXL)             0.21       0.21 f
clk_shift_reg[2]/D (DFFSQXL)             0.00       0.21 f
data arrival time                                   0.21

clock CLK_OUT (rise edge)                0.50       0.50
clock network delay (ideal)              0.00       0.50
clk_shift_reg[2]/CK (DFFSQXL)            0.00       0.50 r
library setup time                      -0.11       0.39
data required time                                  0.39
-----------------------------------------------------------
data required time                                  0.39
data arrival time                                  -0.21
-----------------------------------------------------------
slack (MET)                                        0.18
```

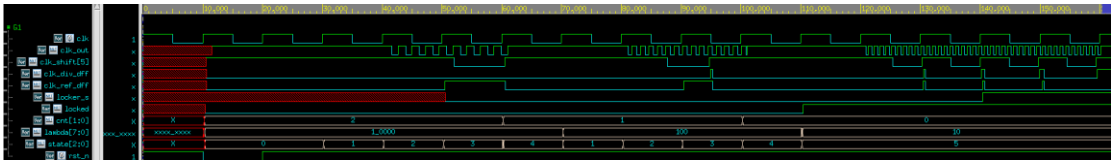
```
Startpoint: state_reg[2]
            (rising edge-triggered flip-flop clocked by CLK_REF)
Endpoint: x_reg[0] (rising edge-triggered flip-flop clocked by CLK_REF)
Path Group: CLK_REF
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock CLK_REF (rise edge)                0.00       0.00
clock network delay (ideal)              0.00       0.00
state_reg[2]/CK (DFFRQX2)                0.00       0.00 r
state_reg[2]/Q (DFFRQX2)                 0.28       0.28 f
U81/Y (CLKINVX1)                         0.06       0.34 r
U70/Y (NAND3XL)                          0.22       0.57 f
U65/Y (AOI211X1)                         0.20       0.76 r
U61/Y (AOI221XL)                         0.13       0.89 f
U54/Y (OAI32XL)                          0.16       1.05 r
x_reg[0]/D (DFFRQX2)                     0.00       1.05 r
data arrival time                                   1.05

clock CLK_REF (rise edge)               10.00      10.00
clock network delay (ideal)              0.00      10.00
x_reg[0]/CK (DFFRQX2)                    0.00      10.00 r
library setup time                      -0.18       9.82
data required time                                  9.82
-----------------------------------------------------------
data required time                                  9.82
data arrival time                                  -1.05
-----------------------------------------------------------
slack (MET)                                        8.77
```

左圖為 clk_out 的 longest path，右圖為 clk_ref 的 longest path。

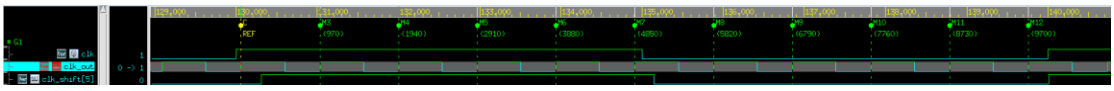
(3) Power consumption

```
  Cell Internal Power  = 181.3881 uW   (80%)
  Net Switching Power  =  44.1792 uW   (20%)
                         ---------
Total Dynamic Power    = 225.5673 uW  (100%)

Cell Leakage Power     =   2.7875 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           0.1296        6.1379e-04          1.0070e+06             0.1313  (  57.48%)
sequential      6.2559e-04        1.2830e-05          6.0107e+04         6.9853e-04  (   0.31%)
combinational   5.1131e-02        4.3553e-02          1.7204e+06         9.6403e-02  (  42.22%)
--------------------------------------------------------------------------------------------
Total              0.1814 mW     4.4179e-02 mW     2.7875e+06 pW         0.2284 mW
```

完成電路合成後，進行模擬，波型圖如下所示：

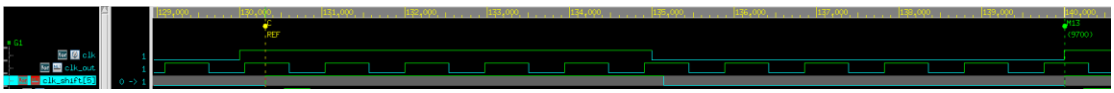


其中 clk 是外部給入的 100MHz 的時脈訊號；clk_out 是 DCO 產生的時脈訊號；clk_shift[5]是 clk_out 除以 10 倍頻的時脈訊號；clk_div_dff 是 clk_shift[5]穿過 PD 第一個 DFF 後的訊號；clk_ref_dff 是 clk 穿過 PD 第一個 DFF 後的訊號；locker_s 是 PD 最後一級做為 locker 功能的 SR-latch 的 S 端訊號(也就是被拿來當作 lead/lag 訊號用)；locked 作為 flag 表示此狀態下產生的 clk_out 已經是 frequency acquisition 後的結果。在 frequency acquisition 後的 clk_out 如下圖：



可以看到週期為 970ps (頻率為 1.03GHz)，與單純只有 DCO 的模擬結果 961ps 相差一點，推測可能是 output stage 所看到的電容不同以及 input driver 的能力不同所造成。10 個 clock cycle 的平均也為 970ps，沒有發生 jitter 的現象。與理想的 clock cycle 1000ps 之 **absolute error 是-30ps，而 relative error 是-3%**。備註：locked 之後的第一個 clk_out 週期即為 970ps。

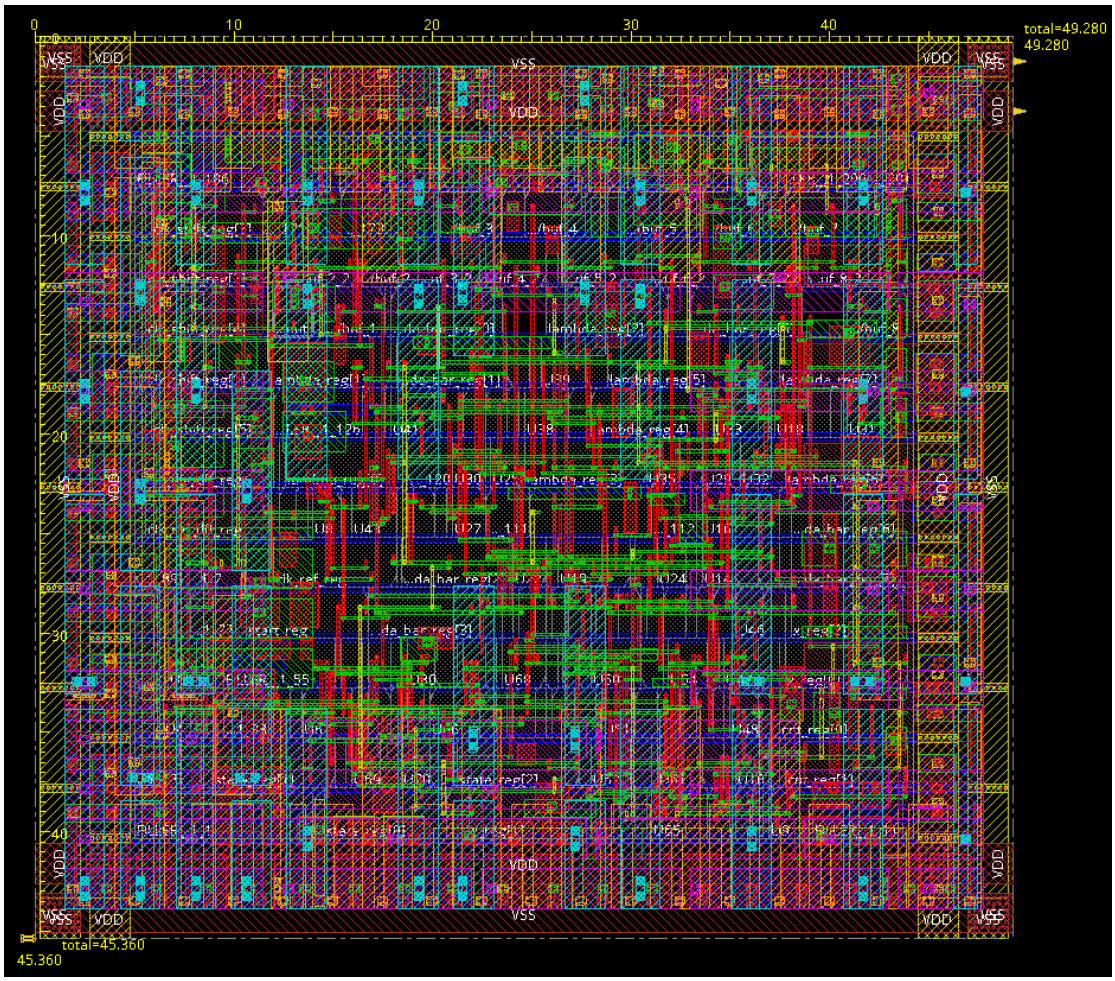
下圖為除頻後的時脈訊號放大圖：



可以看到除頻後的週期為 9700ps，確實為 clk_out 的 10 倍。缺點是與 clk_out 間仍存在約 200ps 的 delay，所幸在這次作業的 resolution 不用到太高。

(c) Try to use SOC Encounter to **generate the layout** of your design. What is the size of your layout? (15%)

APR 的部分這次是直接使用 tcl 檔跑，需要注意的是需要加上 set_dont_touch [get_cells {u_dco}] true 讓 tool 不要優化 DCO 內部的電路，過程中有確認 pre-cts 和 post-cts 的 timing 沒有 violation，geometry、connectivity 和 antenna 的 verification 都沒有錯誤出現。




```
innovus 5> dbGet top.fplan.coreBox_area
1382.976
```

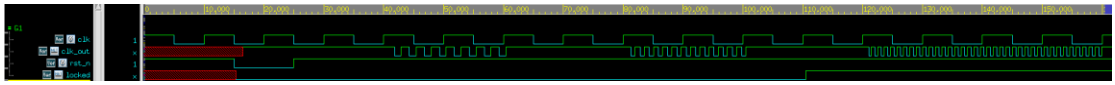
由尺量測之面積為 2235.3408μm^2，由指令得到之 corebox 面積為 1382.976μm^2

Note: power analysis 的結果

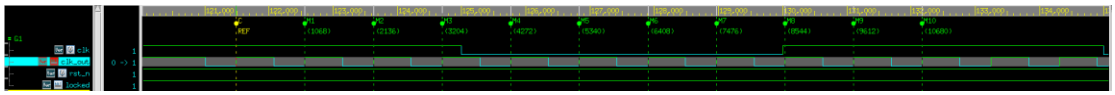
```
Total Power
--------------------------------------------------------------------------------
Total Internal Power:        0.26828180            81.4548%
Total Switching Power:       0.05899209            17.9110%
Total Leakage Power:         0.00208910             0.6343%
Total Power:                 0.32936299
--------------------------------------------------------------------------------
```

(d) Try to do **post-layout Verilog simulation** for your PLL. The netlist should be back-annotated with the post-layout SDF information. Compare your results with those derived in the pre-layout simulation. (15%)

將 innovus 產生的 sdf 檔拿去 back-annotate 整個 PLL 後，得到 simulation 波形：



在 locked 後的放大圖：



可以得知鎖定後 clk_out 之週期為 1.068ns (頻率為 936MHz)，與目標頻率 1GHz 之週期 1ns 的 **absolute error 是 68ps，而 relative error 是 6.8%**。與 pre-sim 時的週期比較，absolute error 是 98ps，而 relative error 是 10.1%。電路功能在 layout 前後都一樣能夠將 clk_out 頻率鎖在靠近 1GHz 的地方，最大的差別在於 clk_out 的週期變長，但也在預期之內，因為在 layout 後，在 DCO 相同的路徑上多考慮了寄生電容與電阻，還有實際接線的 delay，這些都使得其產生的時脈訊號週期變長，其他結果的 layout 前後差別如下表：

	pre-layout	post-layout
Period of clk_out (ns)	0.97	1.068
Frequency of clk_out (GHz)	1.03	0.936
Area (μm^2)	964.56	2235.34
Total Power (mW)	0.2284	0.3294

可以發現 clk_out 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 等等的面積，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，post-layout 的結果在 internal, switching 和 leakage 方面都略有提升，比例沒有變動很多，推測可能是計入更多的寄生電容、電阻使得電能出現更多的消耗。