Lab 5: Synthesizable Asynchronous FIFO

23:59, January 3, 2017

Design Concepts

根據 Clifford E. Cummings 的”Simulation and Synthesis Techniques for Asynchronous FIFO Design” paper 中提供的 style#1 FIFO 為架構，額外加上 style#2 概念中的 almost 以及 error flag，程式碼設計細節如下詳述：

fifo_almost_full_w

由於本身 style#1FIFO 內有利用 wbinext, graynext 來進行判斷下一個 clk 是否指標會繞一圈後讀寫指標只在同一位置，進行輸出 fifo_full_w 訊號的判斷，因此判斷 almost 則再進行運算一次便可計算兩個 clk 後是否會達到 full 便可以達到 almost 訊號的判斷，如下所示：

```
assign alwbinnext = wbinnext + (winc & ~wfull);
assign alwgraynext = (alwbinnext>>1) ^ alwbinnext;
assign alwfull_val =
(alwgraynext=={~wq2_rptr[ADDRSIZE:ADDRSIZE-1], wq2_rptr[ADDRSIZE-2:0]})
||(wgraynext=={~wq2_rptr[ADDRSIZE:ADDRSIZE-1], wq2_rptr[ADDRSIZE-2:0]});
```

fifo_almost_empty_r

almost empty read flag 設計概念如同 write 版本，僅使用 read part 訊號，如下所示：

```
assign alrbinnext = rbinnext + (rinc & ~rempty);
assign alrgraynext = (alrbinnext>>1) ^ alrbinnext;
assign alrempty_val = (alrgraynext == rq2_wptr)||(rgraynext == rq2_wptr);
```

fifo_error_w/ fifo_error_r

而 error flag 則判斷當 full 訊號 asserted，但 winc 也是 asserted，即 buffer 已為滿但仍想寫入 data，則給 error write flag 為 1，read 部分設計概念亦如同 write 版本，如下所示：

```
always @* begin
    if(temp==1 && winc==1)begin fifo_error_w=1;end
    else begin fifo_error_w=0;end
end
```

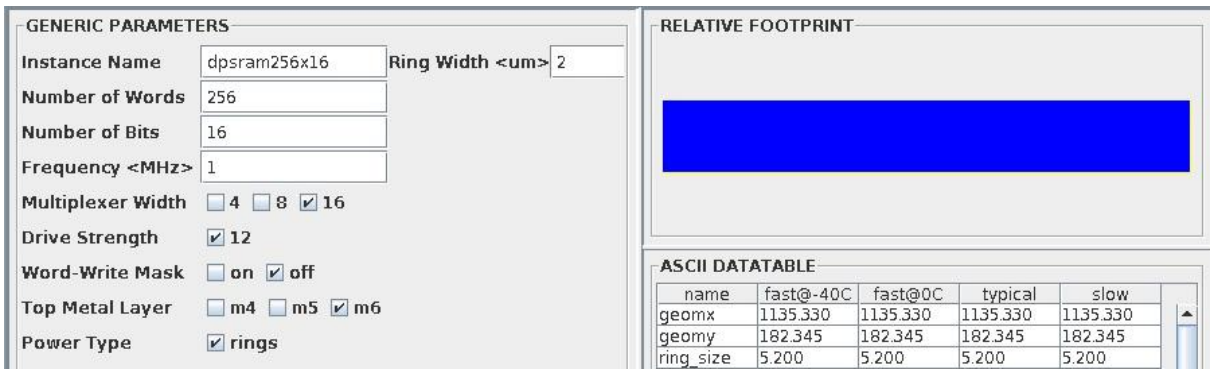
fifo_error_r

```
always @* begin
    if(temp==1 &&rinc==1)begin fifo_error_r=1;end
    else begin fifo_error_r=0;end
end
```

Memory Compiler (Generator)

[iclab79@ic21]/theda21_2/CBDK_TSMC018_Arm_f1.0/CIC/Memory/ra2sh/bin/ra2sh

使用 Artisan 產生需要的 dual-port sram，設定參數如下：



Usage of dpsram256x16

因為要使用 dpsram，因此更改原本 paper 中的 fifomem.v，但因為 sram 本身無法判斷 buffer 是否為滿之後要停止寫入，因此額外使用 CENB 兩條訊號來控制 sram 為 full 時 disable 寫入端，程式碼如下：

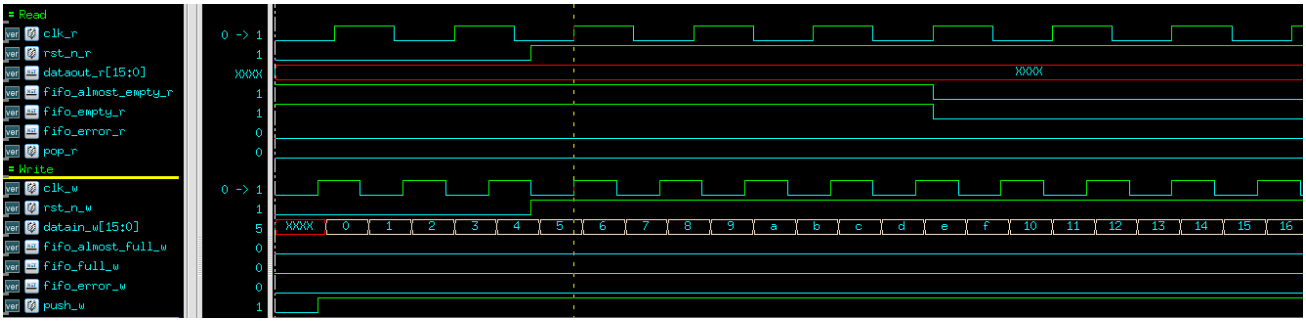
```
assign CENB = (fifo_full_w || !push_w) ? 1 : 0;
```

Stimulation

testbench1 使用相異的 clock domain，clk_w 週期為 10ns，clk_r 週期為 14ns (clk_w < clk_r)。

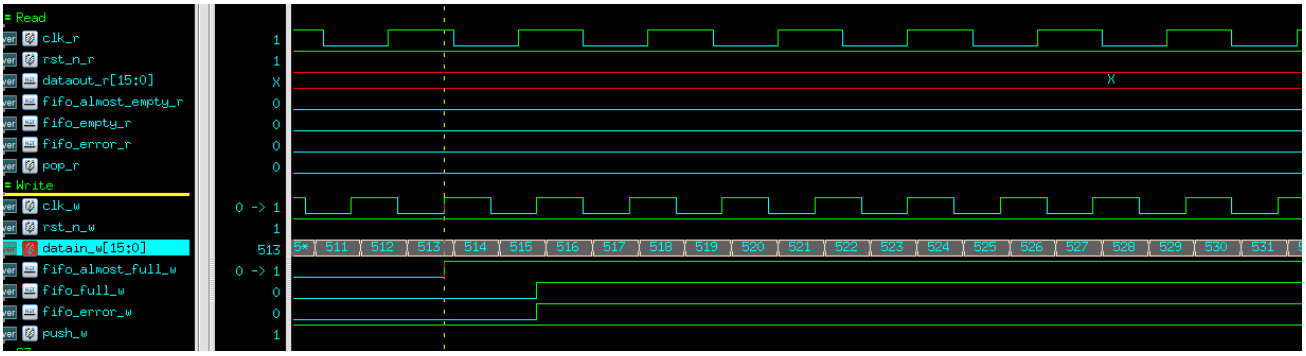
[iclab79@ic21]nWave

1 Write 0 Read – Start writing



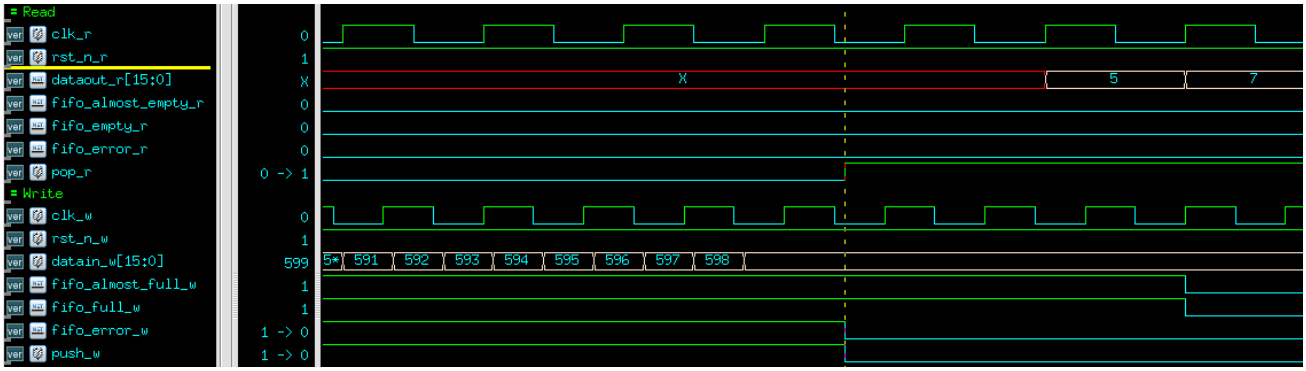
寫指標的控制在 sync_r2w.v 中，因此開始寫入的資料會在 rst_n_w 拉起後的 posedge clk_w 才會開始移動寫入，因此開始寫入的第一個 data 會是 5，並且會寫入每個 5 以後的奇數直到 sram 為滿，此時因為還沒任何資料可以進行 pop out，因此 dataout_t 的輸出都是 unknown。

1 Write 0 Read – Keep pushing data



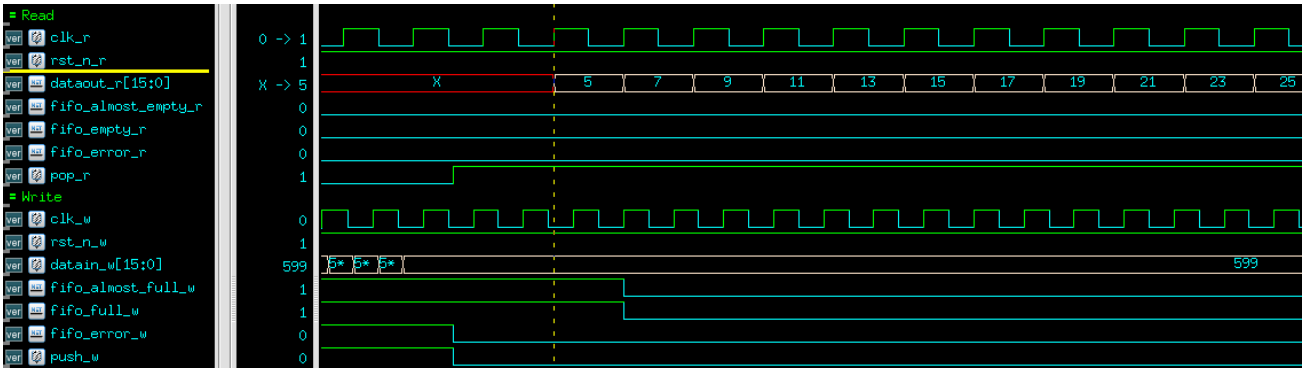
(513-5)/2+1=255，因此在 datain_w 為 513 時 almost_full 會在 clk_w 正緣時被拉起，而下一個 clk_w 因為仍有資料的寫入因此 full 則會在 datain_w 為 515 時被拉起，此時 515 為最後一個 buffer 內的 data，然而因為在 clk_w=515 時仍有 push_w，因此 fifo_error_w 亦會被拉起為 1。

1 Write 0 Read – Stop pushing data



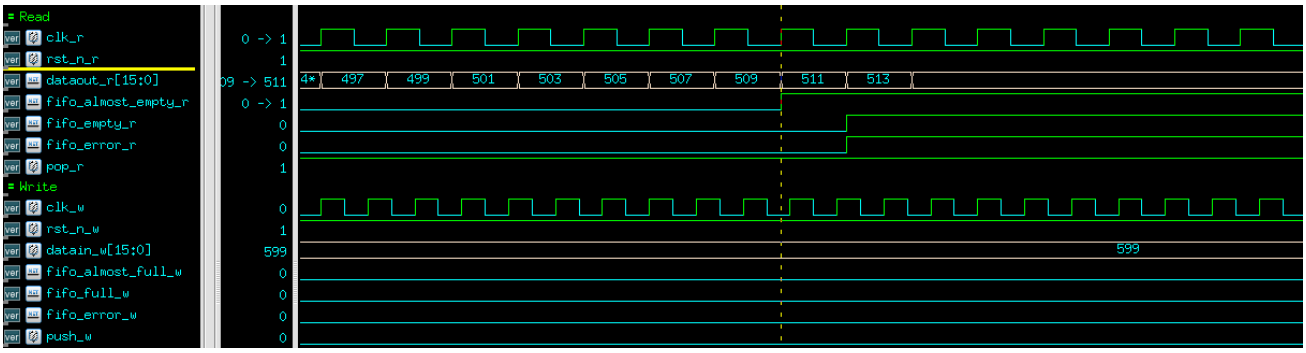
只要 push_w 一掉回 0，此時 fifo_error_w 便會馬上掉回 0，此時 pop 訊號拉起開始進行資料的輸出，開始 pop out 後因為 sram 會在送入位址的下一個 cycle 才會輸出 dataout_r，此時讀指標才會跑到下一個要讀取的資料位址，因此會在輸出第一筆資料後的下一個 posedge clk_r，fifo_almost_full_w 與 fifo_full 才會拉回 0。

0 Write 1 Read – Start reading



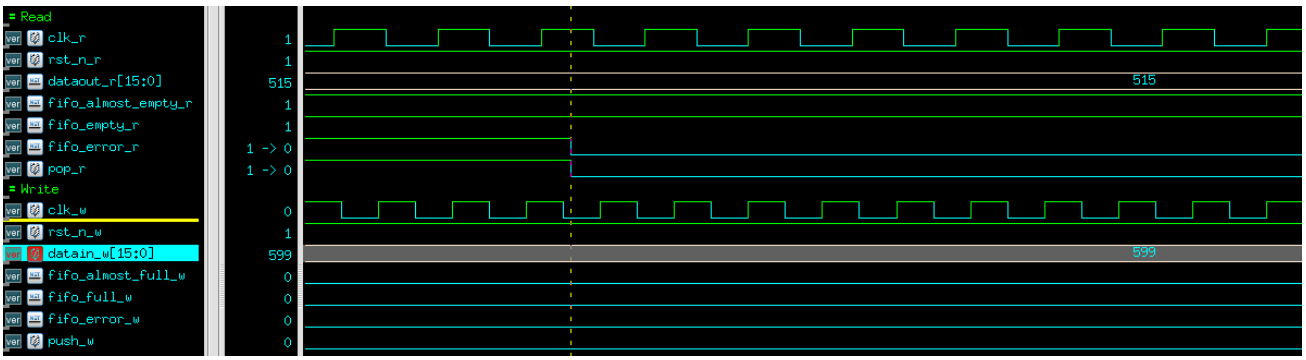
在開始 pop out data 後因為使用 sram 的緣故，dataout_w 會在下一個 clk 輸出，此外因為寫入 data 時從 5 開始寫入，因此輸出會從 5 開始一個個輸出奇數。

0 Write 1 Read –Keep poping data



因為是 sram 因此在 data_out_r 輸出 511 時此時讀指標已經在倒數前一個，因此 alomst_empt 會拉起，而下一個 clk_r 因為繼續 pop out data，因此 fifo_empty_r 會被拉起，而 pop_r 的訊號此時仍為 1，此時 fifo_empty_r 也會被拉起。

0 Write 1 Read – Stop poping data

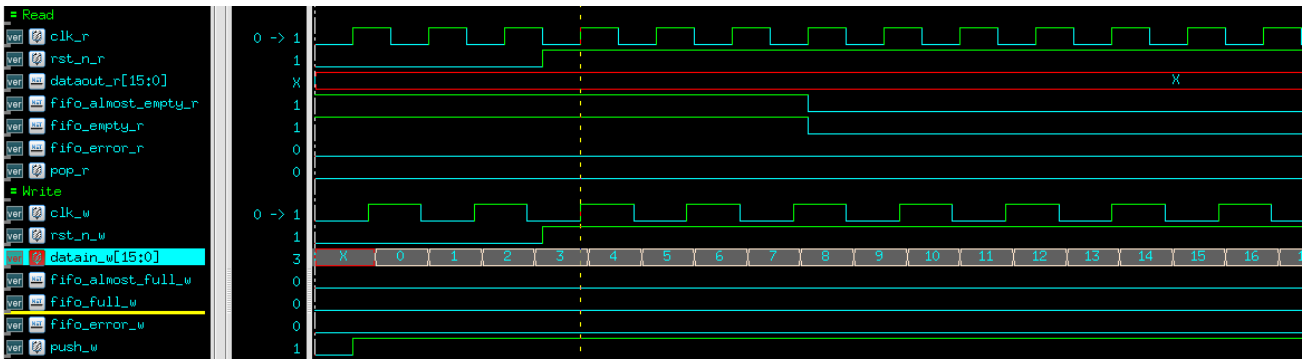


只 pop_r 一掉回 0，此時 fifo_error_r 便會馬上掉回 0，dataout_r 也停在最後一筆輸出的資料 515。

testbench2 使用相異的 clock domain，clk_w 週期為 14ns，clk_r 週期為 10ns (clk_w > clk_r)。

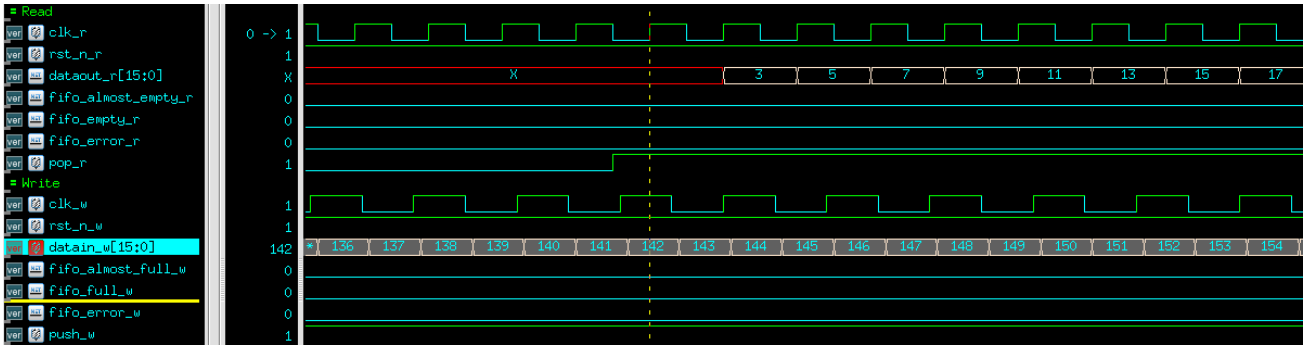
[iclab79@ic21]nWave

1 Write 1 Read – Start writing



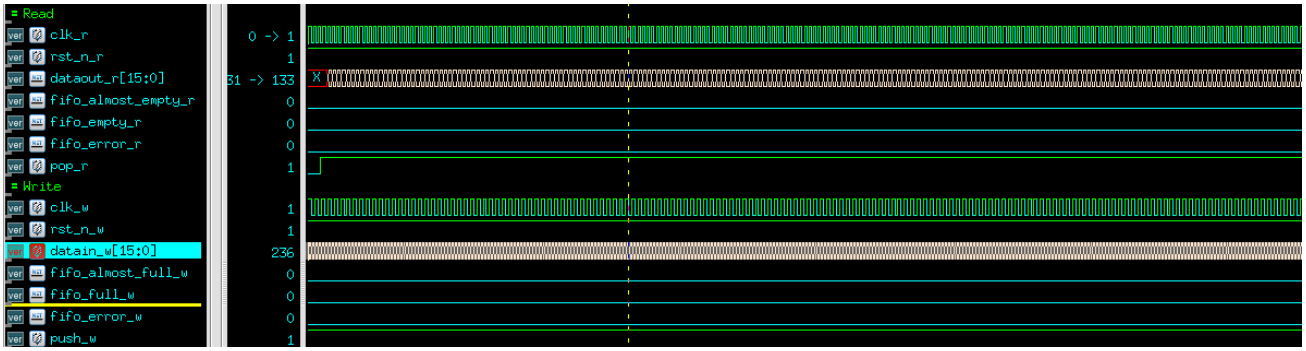
同模擬 1，僅因 clk_w 週期使用的不同，因此第一個寫入的 datain_w 會變為從 3 開始輸入。

1 Write 1 Read – Start reading



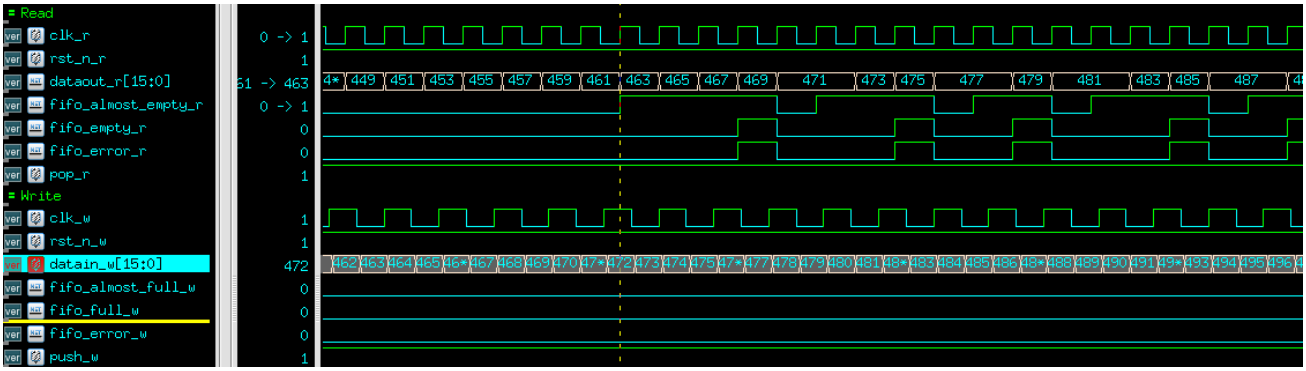
同模擬 1，在 pop_r 拉起後且被 clk_r posedge triggered 後的下一個 clock cycle 才會開始輸出 dataout_r，且從 3 開始輸出。

1 Write 1 Read – Keep reading and writing



因為讀寫同時都會進行，因此在 buffer 沒有滿的情況下可以不斷地進行寫入與讀出，而不會有任何的 almost_empty/full, empty/full, error_w/error_r 訊號的拉起。

1 Write 1 Read – Keep reading and writing (2)



因為讀的週期比較快，因此會追趕上寫入的速度而導致 sram 為空，但因為仍持續的寫入，因此 dataout_w 會繼續保持連續的奇數輸出而不會有影響。

Synthesization

[iclab79@ic21]design_vision

》File》Read

```
Presto compilation completed successfully.
Current design is now '/users/course/2016F/cs5121/iclab79/hw5_test/fifo.db:fifo'
Loaded 5 designs.
Current design is 'fifo'.
design_vision>
Current design is 'fifo'.
Current design is 'fifo'.
```

》File》Analyze

》File》Elabroate

```
Presto compilation completed successfully.
design_vision>
Current design is 'fifo_1'.
Current design is 'fifo'.
```

》Attribute》Specify Clock // for sequential circuits

```
design_vision> create_clock -name "CLK_W" -period 10 -waveform { 0 5  }  { clk_w  }
1
design_vision> set_fix_hold CLK_W
1
design_vision> set_dont_touch_network CLK_W
1

design_vision> create_clock -name "CLK_R" -period 14 -waveform { 0 7  }  { clk_r  }
1
design_vision> set_dont_touch_network CLK_R
1
design_vision> set_fix_hold CLK_R
1
```

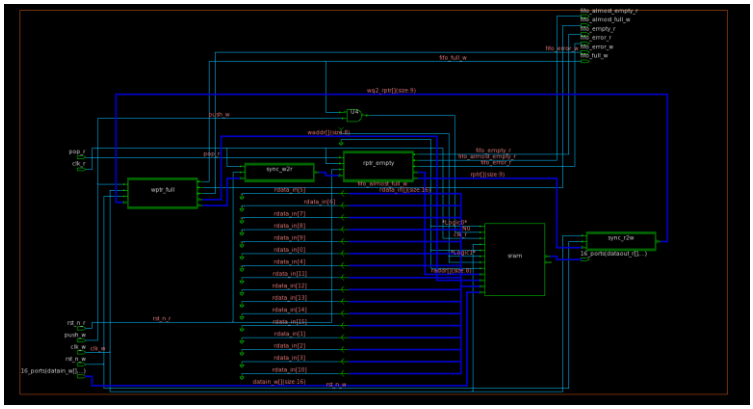
》Attribute》Optimization Constraints》Timing Constraints // for combinational circuits

》Design》Check Design

》Design》Compile Design

```
  Optimization Complete
  ---------------------
1
design_vision>
Current design is 'fifo'.
```

Schematic View



Report Area

```
Number of ports:                          148
Number of nets:                           411
Number of cells:                          274
Number of combinational cells:            165
Number of sequential cells:               104
Number of macros/black boxes:               1
Number of buf/inv:                         10
Number of references:                       6

Combinational area:               1653.267567
Buf/Inv area:                       61.106399
Noncombinational area:            2795.617725
Macro/Black Box area:           220833.734375
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                  225282.619667
Total area:                 undefined
```

Report Power

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	19.0711	0.0000	2.5000e+07	19.0961	(99.14%)	
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	0.1499	2.2463e-03	2.5125e+06	0.1547	(0.80%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	4.8128e-03	3.7330e-03	2.0924e+06	1.0638e-02	(0.06%)	
Total	19.2258 mW	5.9793e-03 mW	2.9605e+07 pW	19.2614 mW		

Report Timing Paths

```
  Point                                    Incr       Path
  -----------------------------------------------------------
  clock CLK R (rise edge)                 28.00      28.00
  clk r (in)                               0.00      28.00 r
  sram/CLKA (dpsram256x16)                 0.00      28.00 r
  data arrival time                                  28.00

  clock CLK W (rise edge)                 30.00      30.00
  clock network delay (ideal)              0.00      30.00
  sram/CLKB (dpsram256x16)                 0.00      30.00 r
  library setup time                      -1.72      28.28
  data required time                                 28.28
  -----------------------------------------------------------
  data required time                                 28.28
  data arrival time                                 -28.00
  -----------------------------------------------------------
  slack (MET)                                         0.28
```

Post-Synthesize Stimulation

RTL 模擬時的 Unix 指令

[iclab79@ic21]ncverilog +nospecify +notimingchecks testbench.v fifo.v dpsram256x18.v +access+r

Gate-Level Post-layout 模擬時的 Unix 指令

[iclab79@ic21]ncverilog testbench.v fifo_syn.v -v /theda21_2/CBDK_TSMC018_\Arm_f1.0/ CIC/Verilog/tsmc18.v +define+SDF +access+r

執行完結果如下:

```
Simulation complete via $finish(1) at time 9 US + 0
./testbench.v:69               #9000 $finish;
ncsim> exit
```

Project Review

在非同步的狀態下要減少錯誤的產生，因此使用格雷碼每次只變動一個 bit 如此在兩個同的 clock domain 才能減少錯誤的機會,此外讀寫指標在送入同步的 sync_w2r/r2w 經過兩層 filp-flop 也讓時間跨到同一 clock domain 再去進行 full/empty 的判斷，是這次可合成異步 FIFO 的重要概念。

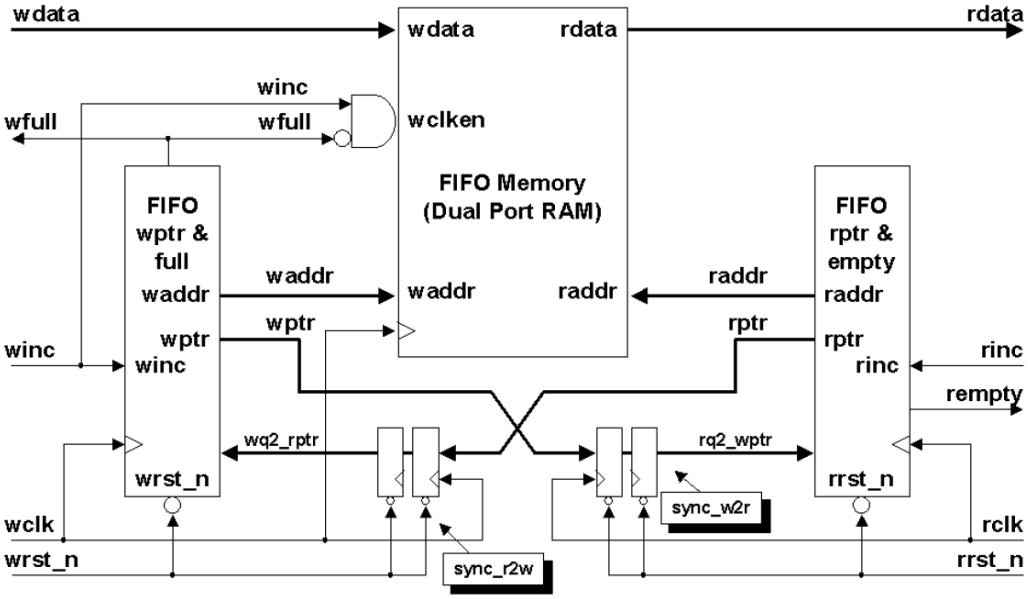


Figure 5 - FIFO1 partitioning with synchronized pointer comparison

這次的 Lab 主要學習到了 memory compiler 以及 library compiler 如何 dump 所需要的 memory 以及 Synopsys Design Compiler 所需要的.db 檔的產生，此外也重新複習了期中 project 所操作過的整個 Design Vision 的使用，特別感謝吳岳騏助教在操作 tool 上讓我打擾很多次，不過也跟助教的討論才得以知道講義上沒有寫的一些參數細節！但最後 slack 雖然有通過 timing 但我推斷是因為一些寫法的錯誤導致沒有辦法順利完成 post synthesize 的模擬，僅希望期末 project 可以順利完成！