CS5121 VLSI System Design and Implementation Fall 2014

Lab 06

# Asynchronous FIFO

103062638 陳煒智

# Description:

使用 Memory compiler 產生出 256 x 16 之 dual-port memory，並使用 FIFO 的方式進行讀取資料。

# Steps:

1. 在 memory compiler 產生 memory 的 RTL code
   產生 RA2SH.v
2. 參考 CummingsSUNG2002SJ_FIFO1.pdf 中的第 6 節的 code
   2.1 記得將 fifo1.v 中的
   fifomen #(DSIZE,ASIZE) fifomem 改成
   RA2SH #(DSIZE,ASIZE) fifomem
   2.2 在 RA2SH 中的接線，重新連接
   新增 almost full, almost empty
3. 先用 ncverilog 去 compile 看看，複製 lab01 的 makefile
   將 SRC 加入所有的 RTL code，包括 RA2SH.v
   直到沒有錯誤，milestone：壓縮成 source.zip
3. 接著，做 RTL simulation 前，先寫個 testbench 測試有無錯誤
4. 將 lib 檔寫入.db 檔中

read_lib RA2SH_slow_syn.lib

read_lib RA2SH_fast@0C_syn.lib

write_lib USERLIB -output fifo_fast.db

write_lib USERLIB -output fifo_slow.db

產生完.db 後，在 synopsys_dc.setup 中加入.db 檔：

set target_library "slow.db fast.db fifo_slow.db fifo_fast.db"

set link_library "slow.db fast.db dw_foundation.sldb fifo_slow.db fifo_fast.db"

5. Synthesis，先用 GUI 做一次流程，包括 analyze, elaborate 及 timing constraints 等等，寫成 TCL file，以便下次直接執行 dc_shell-t –f filename.tcl，即可做 synthesis：

```
analyze -library WORK -format verilog {wptr_full.v sync_w2r.v sync_r2w.v
rptr_empty.v fifo1.v}
elaborate fifo1 -architecture verilog -library DEFAULT
source fifo.dc
uplevel #0 check_design
compile -exact_map
write -hierarchy -format verilog -output fifo_syn.v
write_sdf -version 1.0 -context verilog fifo.sdf
exit
```

6. 接著對 synthesis 後的 fifo_syn.v 加入 testbench 做 simulation


# Report:

Timing report

```
clock rclk' (rise edge)                    10.00      10.00
clock network delay (ideal)                 0.00      10.00
fifomem/CLKB (RA2SH)                        0.00      10.00 r
library setup time                         -0.68       9.32
data required time                                     9.32
-----------------------------------------------------------
data required time                                     9.32
data arrival time                                     -5.48
-----------------------------------------------------------
slack (MET)                                            3.84


clock wclk' (rise edge)                          5.00       5.00
clock network delay (ideal)                      0.00       5.00
fifomem/CLKA (RA2SH)                             0.00       5.00 r
library setup time                              -0.16       4.84
data required time                                          4.84
-----------------------------------------------------------------------
data required time                                          4.84
data arrival time                                          -2.72
-----------------------------------------------------------------------
slack (MET)                                                 2.11
```

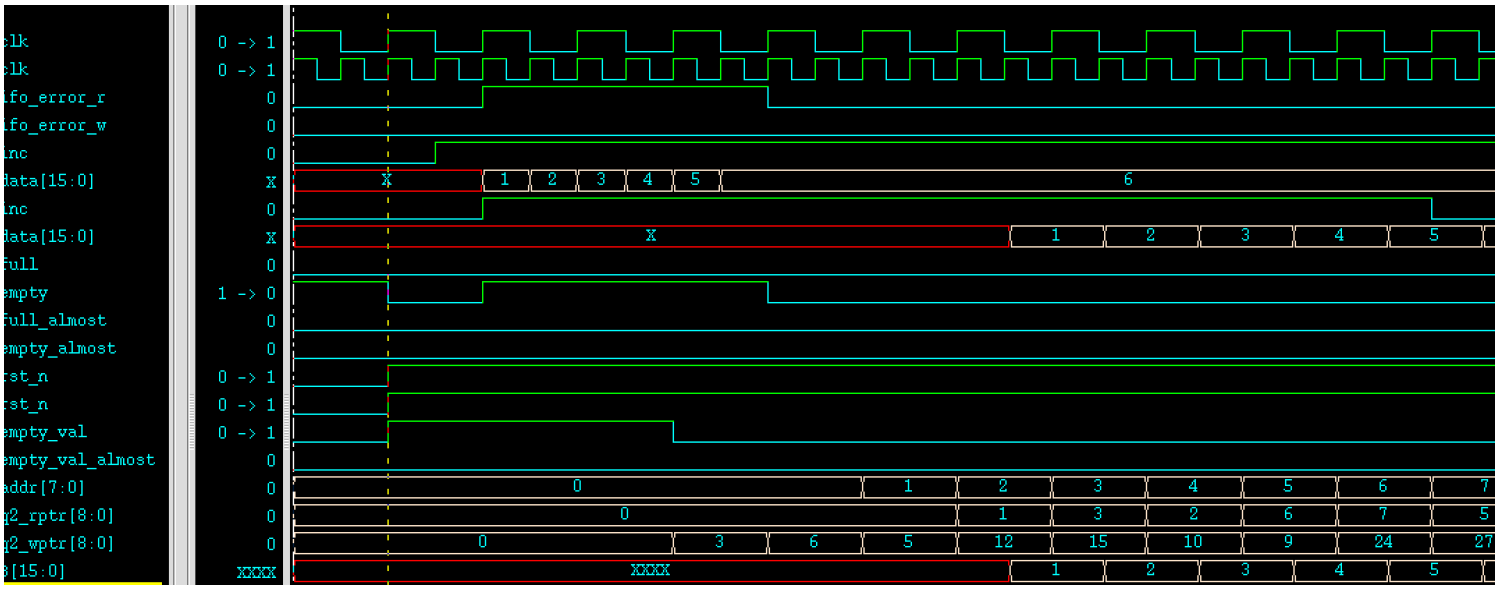
Power report

```
  Cell Internal Power  =  33.3569 mW   (100%)
  Net Switching Power  =  61.2746 uW     (0%)
                          ---------
Total Dynamic Power    =  33.4182 mW   (100%)

Cell Leakage Power     =  14.1824 uW
```

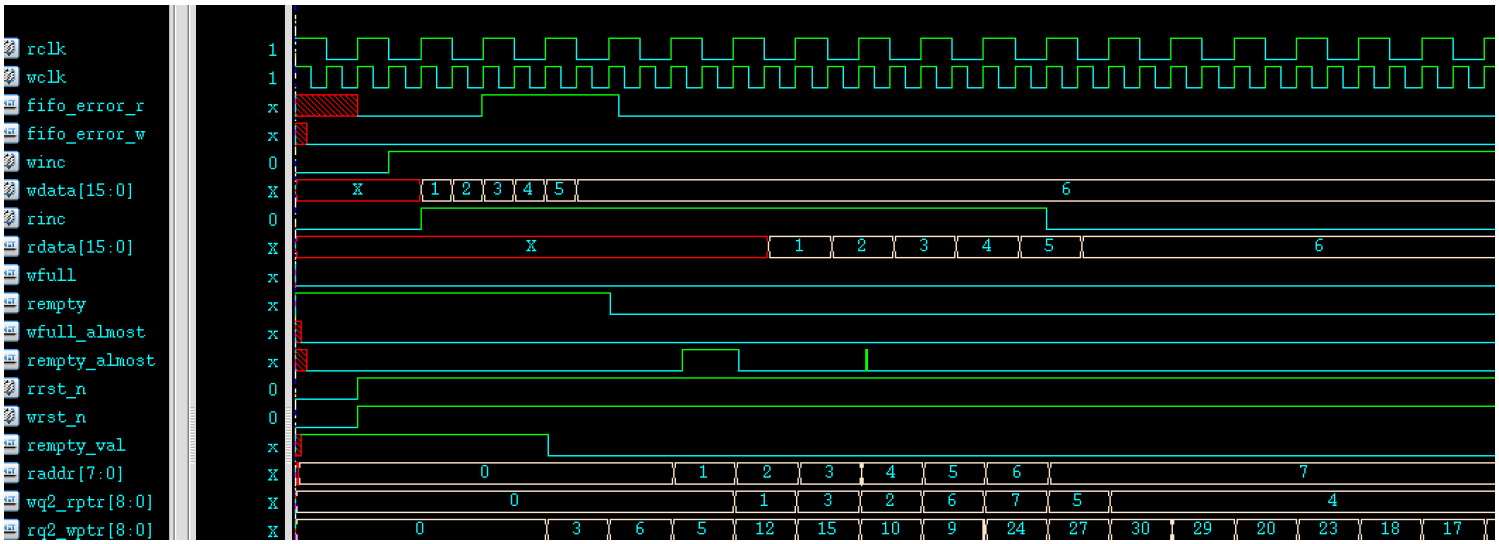
area report

```
Number of ports:                 44
Number of nets:                 143
Number of cells:                 50
Number of references:            14

Combinational area:        3654.502157
Noncombinational area:     223218.581295
Net Interconnect area:       undefined  (No wire load specified)

Total cell area:           226873.083452
Total area:                  undefined
```

# nWave




上圖為 RTL simulation 的 waveform，在 testbench 中，使用 fork join 來同時執行 write 及 read 的測資，一開始沒有寫入任何 data 時，即使 rinc 拉起來，仍然讀不到資料，所以 fifo_error_r 會是 error 拉起來，直到發現有資料了，並且過 2 個 clock cycle(read clock)後(這是因為根據 paper fifo1 的設計，為了處理 clock domain 的問題，加入兩個 FF，所以必須等兩個 cycle)，則 fifo_error_r 會放下為 0，接著再過兩個 read clock 後，資料正式被讀取出來，並且是依照 FIFO 的順序，顯示無誤。




上圖為 synthesis 後的 simulation，其中 rempty_almost 在讀取第二筆資料時，會突然拉起來一下，花了很多時間想把這個問題修掉，但仍然避免不了，除非 rinc 等到寫入完後，才開始 read，才可以解決，但這樣就不符合 dual-port 的設計原則，所以還是無法解決。