N16ADFP -- Logic Synthesis

Click to add subtitle

Lecturer: Yu-Chi Lin

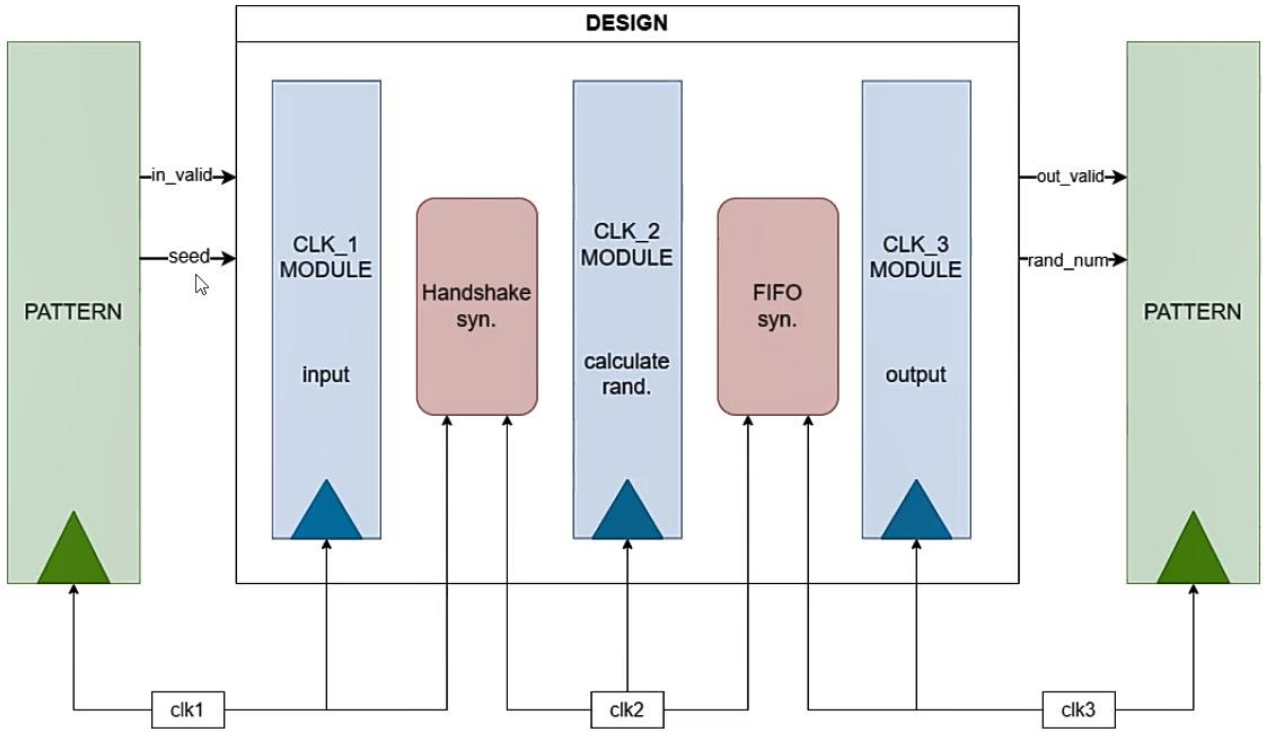
國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Si2 Lab
System Integration
Silicon Implementation

Agenda

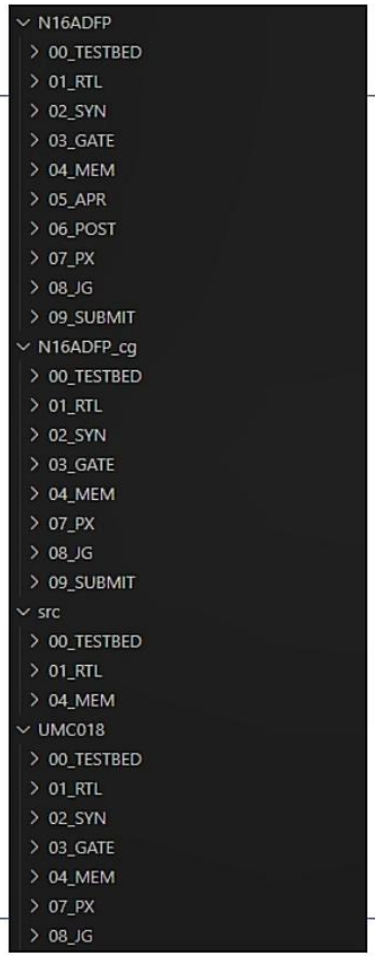
- 01_RTL
- 02_SYN
- 03_GATE
- 04_MEM
- 05_APR
- 06_POST
- 07_PX
- 08_JG

Design


DESIGN
PATTERN
in_valid
seed
CLK_1 MODULE
input
Handshake syn.
CLK_2 MODULE
calculate rand.
FIFO syn.
CLK_3 MODULE
output
out_valid
rand_num
PATTERN
clk1
clk2
clk3

Environment

- **Same design as Lab07**
- **The TESTBED, PATTERN, DESIGN are all given.**
- **Directory**
  - src: storing the design and pattern code
  - UMC018: the working script for 0.18um
  - N16ADFP: the working script for 16nm
  - N16ADFP_cg: the working script for 16nm with clock gating



01_RTL Assertion (Lab07 / Lab10)

- **Use assertion to check synchronizer specification in RTL simulation stage (JG will also check these spec.)**

```
ACK_WO_SREQ: assert property (@(posedge dclk) disable iff (~drst_n) (dack && dreq) |=> dack)
    else $fatal ("ACK_WO_SREQ fail at %t", $time);
DAT_HS_STBL: assert property (@(posedge dclk) disable iff (~drst_n) (dreq && dack) |=> $stable(din))
    else $fatal ("DAT_HS_STBL fail at %t", $time);
NAK_WO_SREQ: assert property (@(posedge dclk) disable iff (~drst_n) (!dack && !dreq) |=> !dack)
    else $fatal ("NAK_WO_SREQ fail at %t", $time);
NRQ_WO_DACK: assert property (@(posedge sclk) disable iff (~srst_n) (!sreq && sack) |=> !sreq)
    else $fatal ("NRQ_WO_DACK fail at %t", $time);
REQ_NO_HOLD: assert property (@(posedge sclk) disable iff (~srst_n) (sreq && !sack) |=> sreq)
    else $fatal ("REQ_NO_HOLD fail at %t", $time);
POP_ON_EMTY: assert property (@(posedge rclk) rinc |-> !rempty)
    else $fatal ("POP_ON_EMTY fail at %t", $time);
PSH_ON_FULL: assert property (@(posedge wclk) winc |-> !wfull)
    else $fatal ("PSH_ON_FULL fail at %t", $time);
RPT_NO_GRAY: assert property (@(posedge rclk) disable iff (~rrst_n) ##1 $changed(rptr) |-> $onehot(rptr ^ $past(rptr)))
    else $fatal ("RPT_NO_GRAY fail at %t", $time);
WPT_NO_GRAY: assert property (@(posedge wclk) disable iff (~wrst_n) ##1 $changed(wptr) |-> $onehot(wptr ^ $past(wptr)))
    else $fatal ("WPT_NO_GRAY fail at %t", $time);
```

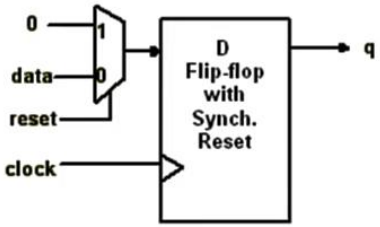
01_RTL Reset Issue (Lab02)

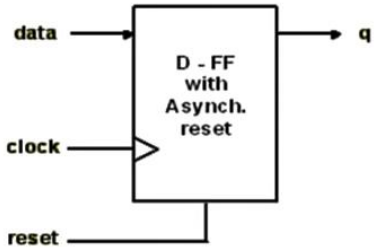
- **Reset**
  - **Synchronous reset**

```
Always @(posedge clk) begin
        if (!rst_n) c <= 0;
        else c <= a+1;
end
```

  - **Asynchronous reset**

```
Always @(posedge clk or negedge rst_n) begin
        if (!rst_n) c <= 0;
        else c <= a+1;
end
```





- **Synchronous reset should meet the setup time and hold time requirement. It is true for all data input.**
- **Asynchronous reset ?**

01_RTL Reset Synchronizer (Lab07)

- The asynchronous reset signal also has timing constraints.
- From sdf (standard delay format) file, you can find out the timing check issue.

```
(CELL
  (CELLTYPE "DFCNQD1BWP16P90LVT")
  (INSTANCE u_FIFO_syn_u_wptr_syn_genblk1_0__u_NDFF_syn/A2_reg)
  (DELAY
    (ABSOLUTE
    (IOPATH (posedge CP) Q (0.034918:0.034957:0.034957) (0.039600:0.039636:0.039636))
    (COND CP == 1'b1 && D == 1'b1 (IOPATH (negedge CDN) Q () (0.025931:0.027182:0.027182)))
    (COND CP == 1'b1 && D == 1'b0 (IOPATH (negedge CDN) Q () (0.025932:0.027183:0.027183)))
    (COND CP == 1'b0 && D == 1'b1 (IOPATH (negedge CDN) Q () (0.026480:0.027795:0.027795)))
    (COND CP == 1'b0 && D == 1'b0 (IOPATH (negedge CDN) Q () (0.026477:0.027789:0.027789)))
    )
  )
  (TIMINGCHECK
    (RECOVERY (posedge CDN) (COND D_SDFCHK (posedge CP)) (0.022947:0.023258:0.023258))
    (HOLD (posedge CDN) (COND D_SDFCHK (posedge CP)) (0.028943:0.028979:0.028979))
    (HOLD (posedge D) (COND CDN_SDFCHK (posedge CP)) (-0.008179:-0.008221:-0.008221))
    (HOLD (negedge D) (COND CDN_SDFCHK (posedge CP)) (0.007975:0.008008:0.008008))
    (SETUP (posedge D) (COND CDN_SDFCHK (posedge CP)) (0.015553:0.015621:0.015621))
    (SETUP (negedge D) (COND CDN_SDFCHK (posedge CP)) (0.001301:0.001332:0.001332))
  )
)
```

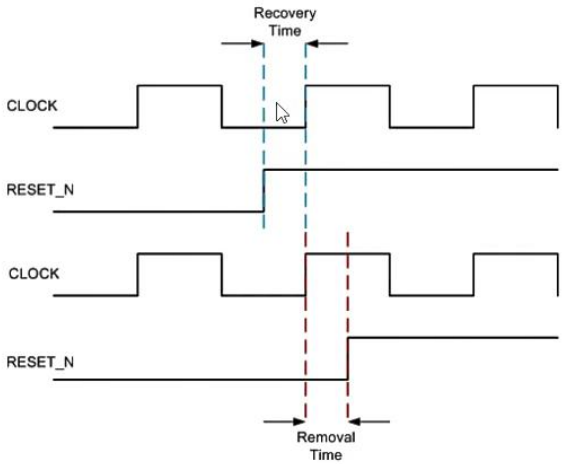
rst clk **N16ADFP**

```
(CELL
  (CELLTYPE "QDFFRBS")
  (INSTANCE u_FIFO_syn_u_rptr_syn/genblk1_7__u_NDFF_syn/A1_reg)
  (DELAY
    (ABSOLUTE
    (IOPATH (posedge CK) Q (0.415228:0.415228:0.415228) (0.411825:0.411825:0.411825))
    (IOPATH (negedge RB) Q () (0.281615:0.281615:0.281615))
    )
  )
  (TIMINGCHECK
    (WIDTH (posedge CK) (0.218920:0.218920:0.218920))
    (WIDTH (negedge CK) (0.435700:0.435700:0.435700))
    (SETUP (posedge D) (posedge CK) (0.203680:0.203680:0.203680))
    (SETUP (negedge D) (posedge CK) (0.134679:0.134679:0.134679))
    (HOLD (posedge D) (posedge CK) (-0.114687:-0.114687:-0.114687))
    (HOLD (negedge D) (posedge CK) (-0.049803:-0.049803:-0.049803))
    (RECOVERY (posedge RB) (posedge CK) (0.062687:0.062687:0.062687))
    (HOLD (posedge RB) (posedge CK) (0.256353:0.256353:0.256353))
    (WIDTH (negedge RB) (0.278040:0.278040:0.278040))
  )
)
```

rst clk **UMC018**

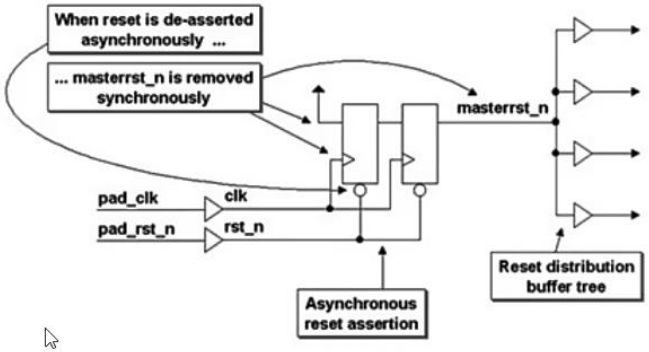
01_RTL Reset Synchronizer (Lab07)

- **Recovery time:**
  - The minimum amount of time that the reset signal (or any other controlling signal) needs to be de-asserted before the arrival of the clock edge.
- **Removal time:**
  - The minimum amount of time that the reset signal (or any other controlling signal) needs to be de-asserted after the arrival of the clock edge.
- **Both recovery time and removal time limit only the de-assertion of the control signal.**
- **Async Reset has asynchronous assert, synchronous de-assert**



01_RTL Reset Synchronizer (Lab07)

- How to solve ? (synchronous de-assertion)
  - Reset synchronizer

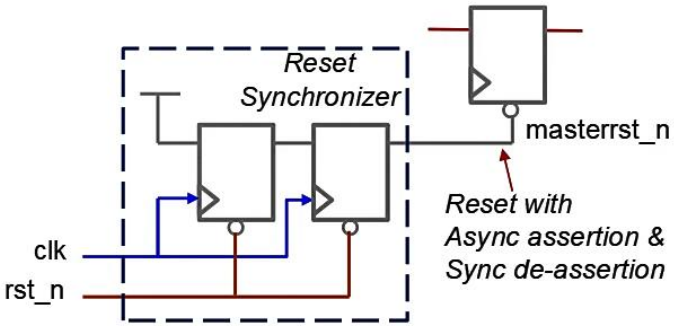


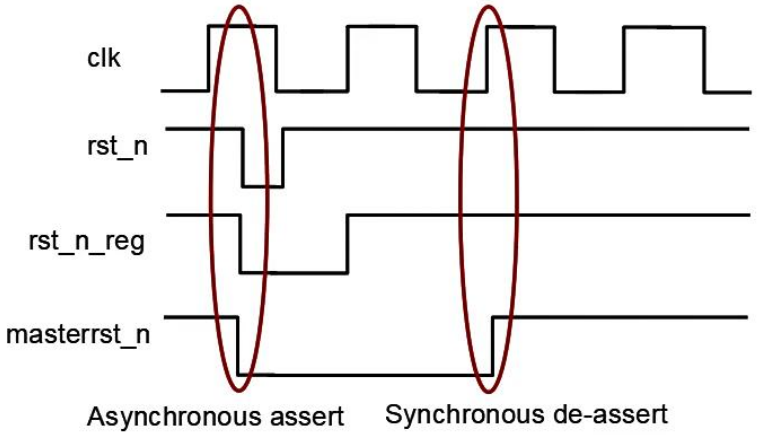
- EVERY ASIC USING ANY ASYNCHRONOUS RESET SHOULD INCLUDE A RESET SYNCHRONIZER CIRCUIT. (not only in CDC design !!)

01_RTL Reset Synchronizer (Lab07)

Think about the metastability issue

- The first flip flop of the reset synchronizer have metastability problems because the input is tied high. The output has been asynchronously reset to 0 and the reset could be de-assert within recovery and removal time of the flip-flop.
- The second flip flop of the reset synchronizer is not subject to metastability because the input and output of the flip-flop are both low when reset is de-asserted.





01_RTL Reset Synchronizer (Lab07)

- In this semester's ICLAB, we force the clock signal to 0 when reset signal is asserted to avoid recovery and removal issue.
- In the code, we add 3 reset synchronizers for clk1, clk2, clk3.
- In pattern, the force – release clk can be removed.
  - ~~force clk=0;~~
  - ~~release clk;~~
- The clock is not force to 0 while asynchronous reset is asserted, the problem is handled by reset synchronizers.

02_SYN Synthesis: Library – stdcell

TSMC ADFP N16 Std-Cell DataBook

Type	Sub-Type	Cell Name Keyword
Combinational	Simple Logic	INV, BUFF, ND, NR, OR, XOR
	Complex Logic	AO, OA, AOI, IAO, IOA, OAI, IND, INR, IIND, IINR, MUX
	Adder	FA1, HA1
	Tie-high / Tie-low Cell	TIEH, TIEL
	Delay Cell	DEL
Storage	Latch	LH
	Flip-Flop	DF
	Scan Flip-Flop	SDC, SDFK, SEDF
Clock Cell	Clock Buffer	CKB, CKN
	Clock And	CKND2, CKAN2
	Clock Or	CKOR2
	Clock Multiplexer	CKMUX2
	Gated Clock Latch	CKLNQ

02_SYN Synthesis: Library – stdcell

TSMC ADFP N16 Std-Cell DataBook

Type	Sub-Type	Cell Name Keyword
Physical Cell	Filler Cell for Core	FILL
	Decoupling Cell	DCAP
	Tap Cell	TAP
	Boundary Cell	BOUN*
Power Management (Multi-supply Voltage)	Simple Logic	PTBUFF, PTINV
	Header	HDR
	Isolation Cell	ISO

02_SYN Synthesis: Library – SRAM (Lab05)

- There is no memory compiler to generate the SRAM, ADFP just gives us some available SRAM to use.

Type	Name	# of words	# bits per word
Single Port SRAM	TS1N16ADFPCLLLVTA128x64M4SWSHOD	128	64
	TS1N16ADFPCLLLVTA16x88M2SWSHOD	16	88
	TS1N16ADFPCLLLVTA16x96M2SWSHOD	16	96
	TS1N16ADFPCLLLVTA512x45M4SWSHOD	512	45
Two port SRAM	TS6N16ADFPCLLLVTA128X32M4FSWHOD	128	32
	TS6N16ADFPCLLLVTA128X64M4FSWHOD	128	64
	TS6N16ADFPCLLLVTA16X120M2FSWHOD	16	120
	TS6N16ADFPCLLLVTA16X32M2FSWHOD	16	32
	TS6N16ADFPCLLLVTA16X72M2FSWHOD	16	72
	TS6N16ADFPCLLLVTA32X32M2FSWHOD	32	32

- We use **TS6N16ADFPCLLLVTA128X32M4FSWHOD** in this lab.

02_SYN Synthesis: Library – SRAM (Lab05)

Single-port SRAM simulation

Signal Name	I/O	Width	Description
CLK	Input	1	Positive edge triggered clock
CEB	Input	1	Chip enable, active low
WEB	Input	1	Write enable, active low
BWEB	Input	Word length (# bits per word)	Write enable bit mask, active low
A	Input	Address width ($log2(# of word))	Address
D	Input	Word length (# bits per word)	Data In
Q	Output	Word length (# bits per word)	Data Out
SLP	Input	1	Sleep (We fixed it to 0)
DSLP	Input	1	Deep Sleep (We fixed it to 0)
SD	Input	1	Shut down (We fixed it to 0)
PUDELAY	Output	1	Delayed signal of SD to inform shut-down process is completed
RTSEL	Input	2	For testing (We fixed it to 2'b01)
WTSEL	Input	2	For testing (We fixed it to 2'b01)

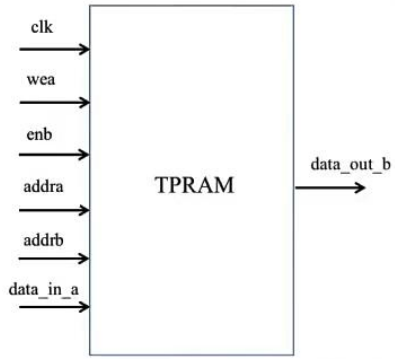
02_SYN Synthesis: Library – SRAM (Lab05)

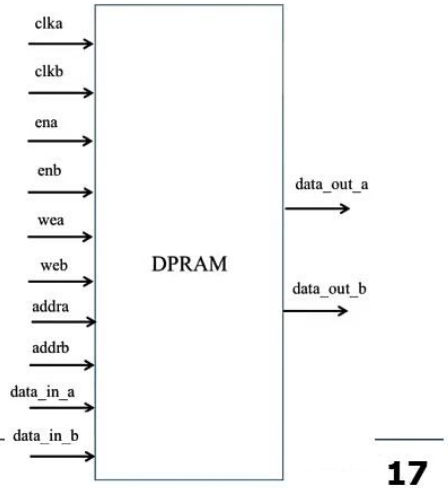
Two-port SRAM simulation

Signal Name	I/O	Width	Description
CLKW	Input	1	Positive edge triggered write clock
AA	Input	Address width ($log2(# of word))	Write address
D	Input	Word length (# bits per word)	Write data
WEB	Input	1	Write enable, active low
BWEB	Input	Word length (# bits per word)	Write enable bit mask, active low
CLKR	Input	1	Positive edge triggered read clock
AB	Input	Address width ($log2(# of word))	Read address
REB	Input	1	Write enable, active low
Q	Output	Word length (# bits per word)	Read data
SLP	Input	1	Sleep (We fixed it to 0)
DSLP	Input	1	Deep Sleep (We fixed it to 0)
SD	Input	1	Shut down (We fixed it to 0)
PUDELAY	Output	1	Delayed signal of SD to inform shut-down process is completed
RTSEL	Input	2	For testing (We fixed it to 2'b01)
WTSEL	Input	2	For testing (We fixed it to 2'b01)
KP	Input	3	For testing (We fixed it to 3'b011)

02_SYN Synthesis: Library – SRAM (Lab05)

- **Two-port SRAM vs. Dual-port SRAM**
  - **Two-port SRAM**
    - 1 read port and 1 write port.
    - Port A is dedicated for read and port B is dedicated for write operation
    - 1 read 0 write / 0 read 1 write / 1 read 1 write
  - **Dual-port SRAM**
    - 2 read ports and 2 write ports.
    - It can be performed independently from both ports if there is no collision
    - 2 read 0 write / 0 read 2 write / 1 read 1 write





02_SYN Synthesis – .synopsys_dc.setup (Lab06)

- Setting stdcell and SRAM library
- Path: /ADFP/Executable_Package/Collaterals/IP/

```
set search_path         " ./ \
                        ../01_RTL                       \
                        ../01_RTL/synchronizer          \
                        ../04_MEM                       \
                        /ADFP/Executable_Package/Collaterals/IP/stdcell/N16ADFP_StdCell/CCS/     \
                        /usr/cad/synopsys/synthesis/cur/libraries/syn/ \
                        /usr/cad/synopsys/synthesis/cur/dw "
set target_library      " N16ADFP_StdCellss0p72vm40c_ccs.db "
set link_library        " * $target_library dw_foundation.sldb N16ADFP_SRAM_ss0p72v0p72vm40c_100a.db "
set dont_use_lst        " TIE* DEL* CKB* CKN* CKMU* G* BHD* DCCK* CK* DEL* *D0* *D18* *D20* *D24* *D48*
*D28* FCICO* ISOLOD1BWP16P90 ISOLOD2BWP16P90 "
set symbol_library      " *.sdb "
set synthetic_library   " dw_foundation.sldb "
```

02_SYN Synthesis – syn.tcl (Lab06, Lab07)

- **Different from the UMC018**
  - **Faster clock period**

```
set CYCLE1 1.41
set CYCLE2 0.49
set CYCLE3 2.07
```

  - **Different design constraints**

```
set_clock_uncertainty 0.027 [all_clocks]
(Please remove set_clock_uncertainty constraints before create CTS in sdc file in 05_APR)
set_driving_cell -lib_cell BUFFD8BWP20P90LVT -pin Z [all_inputs]
set_driving_cell -lib_cell CKBD8BWP20P90LVT -pin Z [all_inputs]
set_load 0.001 [all_outputs]
set_max_fanout 32 [current_design]
(Please remove set_max_fanout constraints in sdc file in 05_APR)
```

  - **Design rule attributes (input transition and capacitance) from the driving cell will be set on the port.**
  - **Solve the timing constraint for recovery and removal path**

```
set enable_recovery_removal_arcs TRUE
```

02_SYN Synthesis – Clock Gating (Lab08)

- Use the function in Design Compiler to automatically clock gating.
- Use the clock gating stdcell in ADFP (CKLNQ*)

```
set compile_clock_gating_through_hierarchy true
set_clock_gating_style -sequential_cell latch \
        -max_fanout 32 \
        -minimum_bitwidth 3 \
        -positive_edge_logic "integrated" \
        -control_point before \
        -control_signal scan_enable \
        -num_stages 4
compile_ultra -gate_clock

report_clock_gating >  Report/$DESIGN\.cg
```

- Only available in N16ADFP_cg

Clock Gating Summary

Number of Clock gating elements	11
Number of Gated registers	185 (64.46%)
Number of Ungated registers	102 (35.54%)
Total number of registers	287

Clock Gating Report by Origin

	Actual (%) Count
Number of tool-inserted clock gating elements	11 (100.00%)
Number of pre-existing clock gating elements	0 (0.00%)
Number of gated registers	185 (64.46%)
Number of tool-inserted gated registers	185 (64.46%)
Number of pre-existing gated registers	0 (0.00%)
Number of ungated registers	102 (35.54%)
Number of registers	287

02_SYN Synthesis – PRGN_TOP.sdc / pt.tcl (Lab07)

- **The design constraints for CDC and RDC design in synthesis stage.**

```
set_clock_groups -name group1 -asynchronous -group {clk1} -group {clk2} -group {clk3}
set_false_path -from [get_ports rst_n] -to [all_clocks]
```

- **Remove timing check using PrimeTime for gate level simulation.**

```
foreach_in_collection x [get_cell */A1_reg] {
  set_annotated_check -0 -setup -from [get_object_name $x]/CP -to [get_object_name $x]/D -clock rise
  set_annotated_check -0 -hold  -from [get_object_name $x]/CP -to [get_object_name $x]/D -clock rise
}

foreach_in_collection x [get_cell */B1_reg] {
  set_annotated_check -0 -recovery -from [get_object_name $x]/CP -to [get_object_name $x]/CDN -clock rise
  set_annotated_check -0 -removal  -from [get_object_name $x]/CP -to [get_object_name $x]/CDN -clock rise
}
```

02_SYN Synthesis – PRGN_TOP.sdc / pt.tcl (Lab07)

After PrimeTime, In the sdf file

```
(CELL
  (CELLTYPE "DFCNQD1BWP16P90LVT")
  (INSTANCE u_Handshake_syn_u_ack_syn/A1_reg)
  (DELAY
    (ABSOLUTE
    (IOPATH (posedge CP) Q (0.033::0.033) (0.039::0.039))
    (COND CP==1'b1&&D==1'b1  (IOPATH (negedge CDN) Q () (0.031::0.033)))
    (COND CP==1'b0&&D==1'b0  (IOPATH (negedge CDN) Q () (0.031::0.034)))
    (COND CP==1'b0&&D==1'b1  (IOPATH (negedge CDN) Q () (0.031::0.034)))
    (COND CP==1'b1&&D==1'b0  (IOPATH (negedge CDN) Q () (0.031::0.033)))
    )
  )
  (TIMINGCHECK
    (WIDTH    (COND CDN_D_SDFCHK  (posedge CP)) (0.016::0.016))
    (WIDTH    (COND CDN_nD_SDFCHK  (posedge CP)) (0.019::0.019))
    (WIDTH    (COND CDN_D_SDFCHK  (negedge CP)) (0.023::0.023))
    (WIDTH    (COND CDN_nD_SDFCHK  (negedge CP)) (0.023::0.023))
    (HOLD (posedge CDN)    (COND D_SDFCHK  (posedge CP)) (0.031::0.031))
    (RECOVERY (posedge CDN)    (COND D_SDFCHK  (posedge CP)) (0.021::0.021))
    (WIDTH    (COND CP_D_SDFCHK  (negedge CDN)) (0.016::0.016))
    (WIDTH    (COND CP_nD_SDFCHK  (negedge CDN)) (0.016::0.016))
    (WIDTH    (COND nCP_D_SDFCHK  (negedge CDN)) (0.010::0.010))
    (WIDTH    (COND nCP_nD_SDFCHK  (negedge CDN)) (0.010::0.010))
    (SETUP D    (COND CDN_SDFCHK  (posedge CP)) (-0.000::-0.000))
    (HOLD D    (COND CDN_SDFCHK  (posedge CP)) (-0.000::-0.000))
  )
)
```

```
(CELL
  (CELLTYPE "DFCNQD1BWP16P90LVT")
  (INSTANCE u_rst3_syn/B1_reg)
  (DELAY
    (ABSOLUTE
    (IOPATH (posedge CP) Q (0.033::0.033) (0.039::0.039))
    (COND CP==1'b1&&D==1'b1  (IOPATH (negedge CDN) Q () (0.012::0.013)))
    (COND CP==1'b0&&D==1'b0  (IOPATH (negedge CDN) Q () (0.012::0.013)))
    (COND CP==1'b0&&D==1'b1  (IOPATH (negedge CDN) Q () (0.012::0.013)))
    (COND CP==1'b1&&D==1'b0  (IOPATH (negedge CDN) Q () (0.012::0.013)))
    )
  )
  (TIMINGCHECK
    (WIDTH    (COND CDN_D_SDFCHK  (posedge CP)) (0.016::0.016))
    (WIDTH    (COND CDN_nD_SDFCHK  (posedge CP)) (0.019::0.019))
    (WIDTH    (COND CDN_D_SDFCHK  (negedge CP)) (0.023::0.023))
    (WIDTH    (COND CDN_nD_SDFCHK  (negedge CP)) (0.023::0.023))
    (HOLD (posedge CDN)    (COND D_SDFCHK  (posedge CP)) (-0.000::-0.000))
    (RECOVERY (posedge CDN)    (COND D_SDFCHK  (posedge CP)) (-0.000::-0.000))
    (WIDTH    (COND CP_D_SDFCHK  (negedge CDN)) (0.016::0.016))
    (WIDTH    (COND CP_nD_SDFCHK  (negedge CDN)) (0.016::0.016))
    (WIDTH    (COND nCP_D_SDFCHK  (negedge CDN)) (0.010::0.010))
    (WIDTH    (COND nCP_nD_SDFCHK  (negedge CDN)) (0.010::0.010))
    (SETUP (posedge D)    (COND CDN_SDFCHK  (posedge CP)) (0.015::0.015))
    (SETUP (negedge D)    (COND CDN_SDFCHK  (posedge CP)) (0.002::0.002))
    (HOLD (posedge D)    (COND CDN_SDFCHK  (posedge CP)) (-0.006::-0.006))
    (HOLD (negedge D)    (COND CDN_SDFCHK  (posedge CP)) (0.010::0.010))
  )
)
```

02_SYN Synthesis – Area

- You can observe the timing and area improvement by comparing the result of synthesis of UMC018 and N16ADFP

```
Library(s) Used:

    fsa0m_a_generic_core_ss1p62v125c (File: /RAID2/COURSE/iclab/iclabTA01/UMC018_CBDK/CIC/SynopsysDC/db/fsa0m_a_generic_core_ss1p62v125c.db)
    SJMA180_128X32X1BM1_WC (File: /RAID2/COURSE/iclab/iclabTA02/Desktop/BONUS_ADFP_v3/UMC018/04_MEM/SJMA180_128X32X1BM1_WC.db)

Number of ports:                        494
Number of nets:                        1305
Number of cells:                        860
Number of combinational cells:          546
Number of sequential cells:             286
Number of macros/black boxes:             1
Number of buf/inv:                      100
Number of references:                    40

Combinational area:             8983.800025
Buf/Inv area:                    965.563190
Noncombinational area:         17048.908913
Macro/Black Box area:         198684.421875
Net Interconnect area:     undefined  (Wire load has zero net area)

Total cell area:              224717.130813
Total area:                undefined
```

~60X

```
Library(s) Used:

    N16ADFP_StdCellss0p72vm40c_ccs (File: /ADFP/Executable_Package/Collaterals/IP/stdcell/N16ADFP_StdCell/CCS/N16ADFP_StdCellss0p72vm40c_ccs.db)
    N16ADFP_SRAM_ss0p72v0p72vm40c (File: /ADFP/Executable_Package/Collaterals/IP/sram/N16ADFP_SRAM/NLDM/N16ADFP_SRAM_ss0p72v0p72vm40c_100a.db)

Number of ports:                        151
Number of nets:                         932
Number of cells:                        798
Number of combinational cells:          489
Number of sequential cells:             285
Number of macros/black boxes:             1
Number of buf/inv:                       57
Number of references:                    57

Combinational area:              177.552003
Buf/Inv area:                      9.745920
Noncombinational area:           326.177292
Macro/Black Box area:           3202.053711
Net Interconnect area:     undefined  (Wire load has zero net area)

Total cell area:                3705.783006
Total area:                undefined
```

03_GATE Simulation (Lab03)

- **The result after synthesis still remain hold time violation.**
- **It is expected because the hold time issue is handled in APR stage and the advance process has more serious hold time issue.**
- **There is no function provided by VCS to just check the setup time and ignore the hold time violation**
  - Use PrimeTime to remove all hold time check
  - Write custom function to ignore it while simulation
- **Recall… PLI (Programming Language Interface) function**
  - We use PLI function to dump fsdb file. ($fsdbDumpfile)
  - In makefile, we include the PLI function library (static library).
  - Write your own code to implement the PLI function
    (You can find the c code in src/00_TESTBED)
  - Use PLI function in TESTBED.v

```
VCS_GATE_SIM = vcs ${TIMESCALE} \
    -j${num_CPU_cores} \
    -sverilog \
    +v2k \
    -full64 \
    -Mupdate \
    -R \
    -debug_access+all \
    -f ${source_file} \
    -o ${output_file} \
    -l ${log_file} \
    -P ${VERDI}/share/PLI/VCS/linux64/novas.tab \
       ${VERDI}/share/PLI/VCS/linux64/pli.a \
    -P ../00_TESTBED/NoTimingChecks.tab \
       ../00_TESTBED/NoTimingChecks.a \
    -v ${N16ADFP_SIM} \
    +define+GATE \
    +define+${define} \
    +neg_tchk \
    +nowarnNTCDSN
```