2024 Memory Compiler(40nm)&Vivado BRAM

EC5015－VLSI Lab

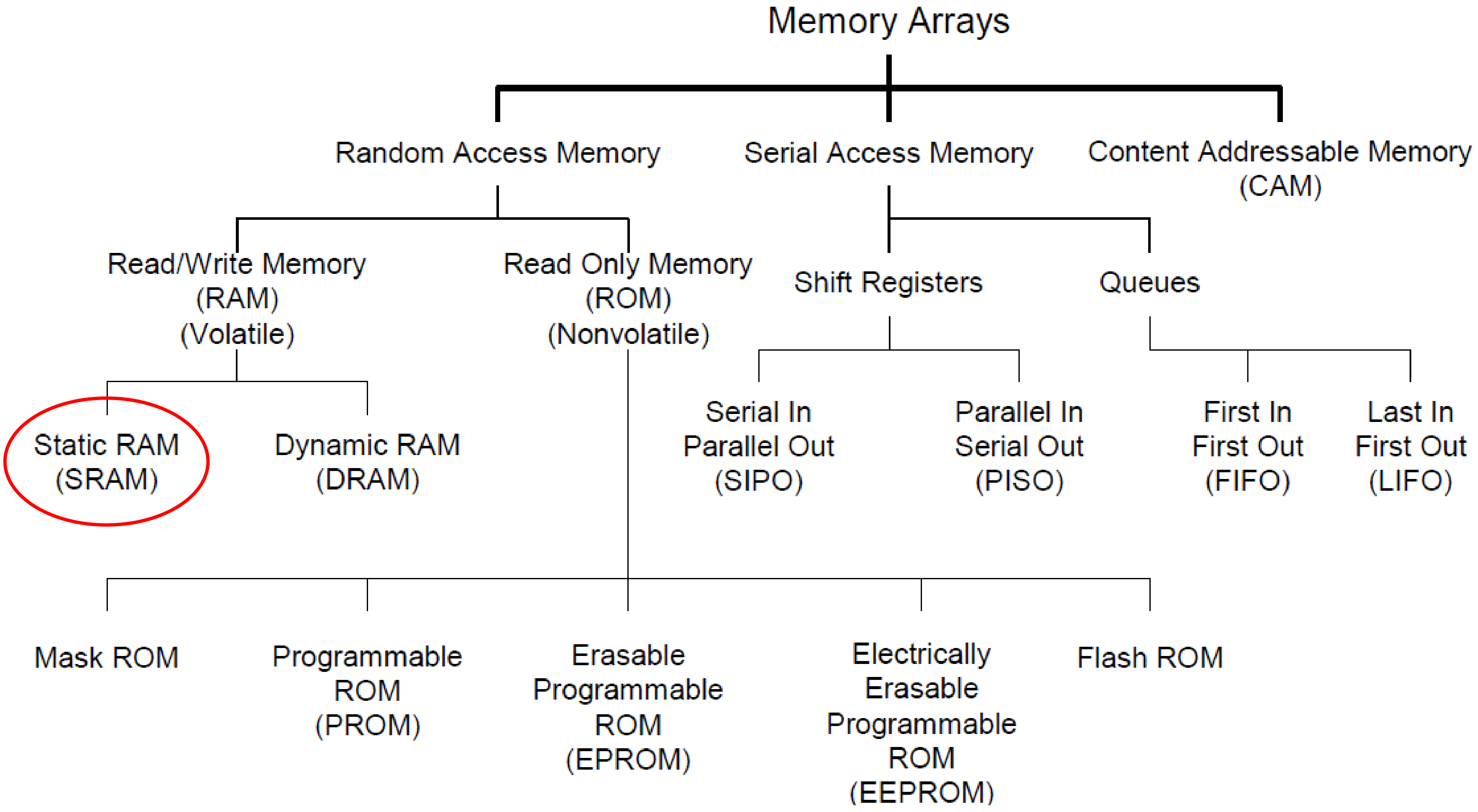
王鼎睿 rueimail4work@g-mail.nsysu.edu.tw

Outline

- Type of memory arrays
- ARM memory compiler introduction
- ARM memory parameter
- Memory Compiler Flow
- Memory lib to db Flow
- Memory Pre-sim Flow
- Memory Synthesis Flow
- Memory Gate-level-Simulation Flow
- Vivado BRAM

Memory Compiler

Type of memory arrays


Memory Arrays
Random Access Memory
Serial Access Memory
Content Addressable Memory (CAM)
Read/Write Memory (RAM) (Volatile)
Read Only Memory (ROM) (Nonvolatile)
Shift Registers
Queues
Static RAM (SRAM)
Dynamic RAM (DRAM)
Serial In Parallel Out (SIPO)
Parallel In Serial Out (PISO)
First In First Out (FIFO)
Last In First Out (LIFO)
Mask ROM
Programmable ROM (PROM)
Erasable Programmable ROM (EPROM)
Electrically Erasable Programmable ROM (EEPROM)
Flash ROM

ARM memory compiler introduction（1/5）

- 由ARM公司提供

- 常用的Memory種類

TSMC_40nm	Register file	SRAM
Single-port	RF_SP_HDE (rvt_hvt_rvt) RF_SP_HSD (rvt_rvt_hvt)	SRAM_SP_HDE (rvt_hvt_rvt) SRAM_SP_HSC (rvt_hvt_rvt)
Two-port	RF_2P_HSE (rvt_hvt_rvt)	-
Dual-port	--	SRAM_DP_HDE (rvt_hvt_rvt)

ARM memory compiler introduction（2/5）

- Register file
  - 容量較小
  - 只支援單一讀或寫的埠(1R/1W、1R1W)
  - 相同容量下面積較小

- SRAM
  - 容量較大
  - 可支援兩個讀及寫的埠(1R/1W、1R1W/2R/2W)
  - 相同容量下面積較大

ARM memory compiler introduction（3/5）

- Single-Port
  - 同一時間只能做單端讀取(1R)或是單端寫入(1W)的功能

Pin	Description
CEN	Chip Enable (active low)
WEN WEN=0 write ; WEN=1 read	Write Enable (active low)
A	Addresses(A[0]=LSB)
D	Data Inputs (D[0]=LSB)
Q	Data Outputs (Q[0]=LSB)
CLK	Clock
WENY	Multiplexor out (WEN CEN A D)
CENY	
AY DY	
EMA	Extra Margin Adjustment
EMAW	
EMAS	

Pin	Description
BEN	Bypass mode, active low
TEN (enable)	Test Mode Enable ,active low
TCEN	Chip Enable Test Input ,active low
TWEN	Write Enable Test Input ,active low
TA	Addresses Test Input(TA[0]=LSB)
TD	Data Test Inputs (TD[0]=LSB)
TQ	Bypass Q input in write mode(TQ[0] = LSB)
RET1N	Retention mode, active low
STOV	Synchronous clock enable,

ARM memory compiler introduction（4/5）

- Dual-Port
  - 同一時間兩個埠都可做讀取或寫入的功能(1R/1W/1R1W/2R/2W)

Pin	Description
CLKA CLKB	Port A&B Clocks
CENA CENB	Port A&B Chip Enables(Active low)
WENA WENB	Port A&B Write Enables(Active low) WEN=0 write ; WEN =1 read
AA AB	Port A&B Addresses (AA[0],AB[0]=LSB)
DA DB	Port A&B Data Inputs (DA[0],DB[0]=LSB)
QA QB	Port A&B Data Outputs (QA[0],QB[0]=LSB)
EMAA EMAWA EMASA	Read & Wrtie Extra Margin Adjustment
EMAB EMAWB EMASB	
RET1N	Retention mode, acitve low
STOVA STOVB	Synchronous clock enable,active high

Pin	Description
AYA AYB	Multiplexor out (ADDR DATA_IN CEN)
DYA DYB	
CENYA CENYB	
WENYA WENYB	
BENA BENB TQA TQB	Bypass mode,active low (Bypass mode_EN data_in)
TENA TENB	TEST MODE,active low (TEST_MODE_EN CEN ADDR DATA_IN)
TCENA TCENB TWENA TWENB	
TAA TAB	
TDA TDB	
COLLDISN	Collision circuit disable,active low

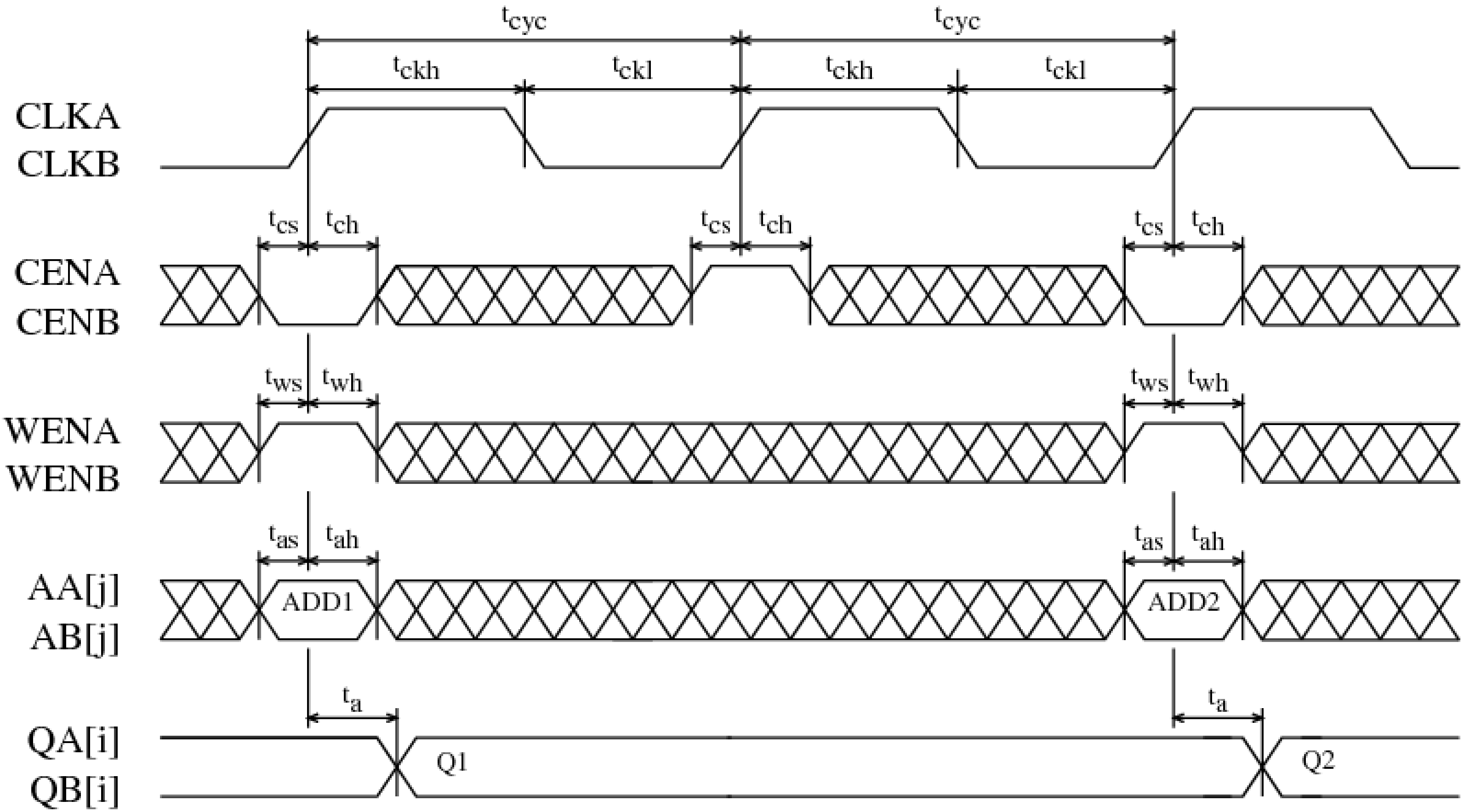
ARM memory compiler introduction（5/5）

- Two-Port Port A is Read only Port B is Write only → 因此沒有WEN port
  - 同一時間只能做單端讀取及單端寫入的功能(1W/1R/1R1W)

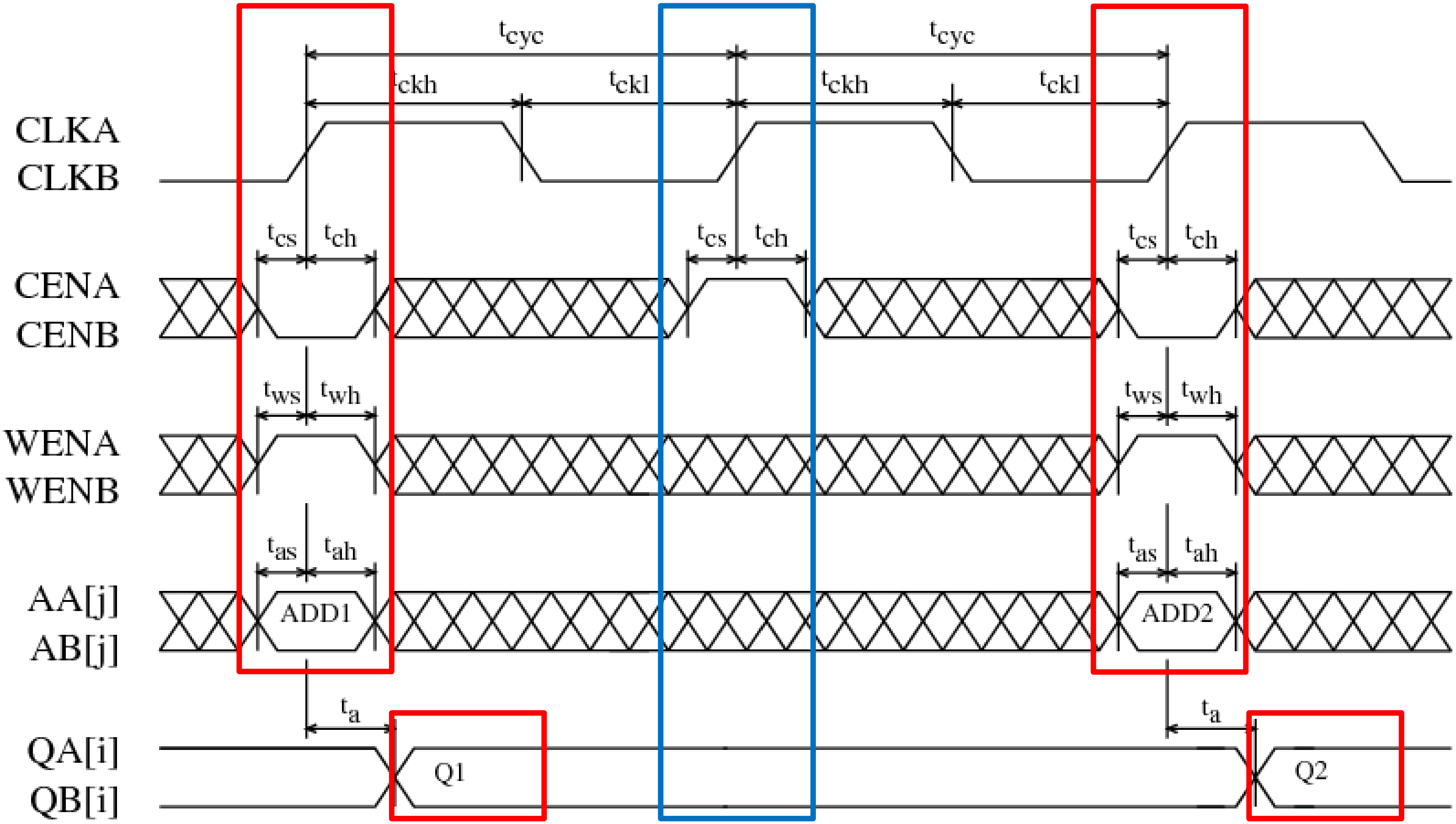
Pin	Description
CLKA CLKB	Read & Write Clocks
CENA CENB	Read & Write Enables (Active low)
AA AB	Read & Write Addresses (AA[0],AB[0]=LSB)
DB	Data Inputs (DB[0]=LSB)
QA	Data Outputs (QA[0]=LSB)
EMAA EMASA	Read & Wrtie Extra Margin Adjustment
EMAB EMAWB	
COLLDISN	Collision circuit disable,active low
RET1N	Retention mode, acitve low
STOVA	Synchronous clock enable,active high

Pin	Description
AYA AYB	Multiplexor out (ADDR DATA_IN CEN)
DYB	
CENYA CENYB	
BENA TQA	Bypass mode,active low (Bypass mode_EN data_in)
TENA TENB	TEST MODE,active low (TEST_MODE_EN CEN ADDR DATA_IN)
TCENA TCENB	
TAA TAB	
TDB	

Read Cycle Timing(Dual Port)

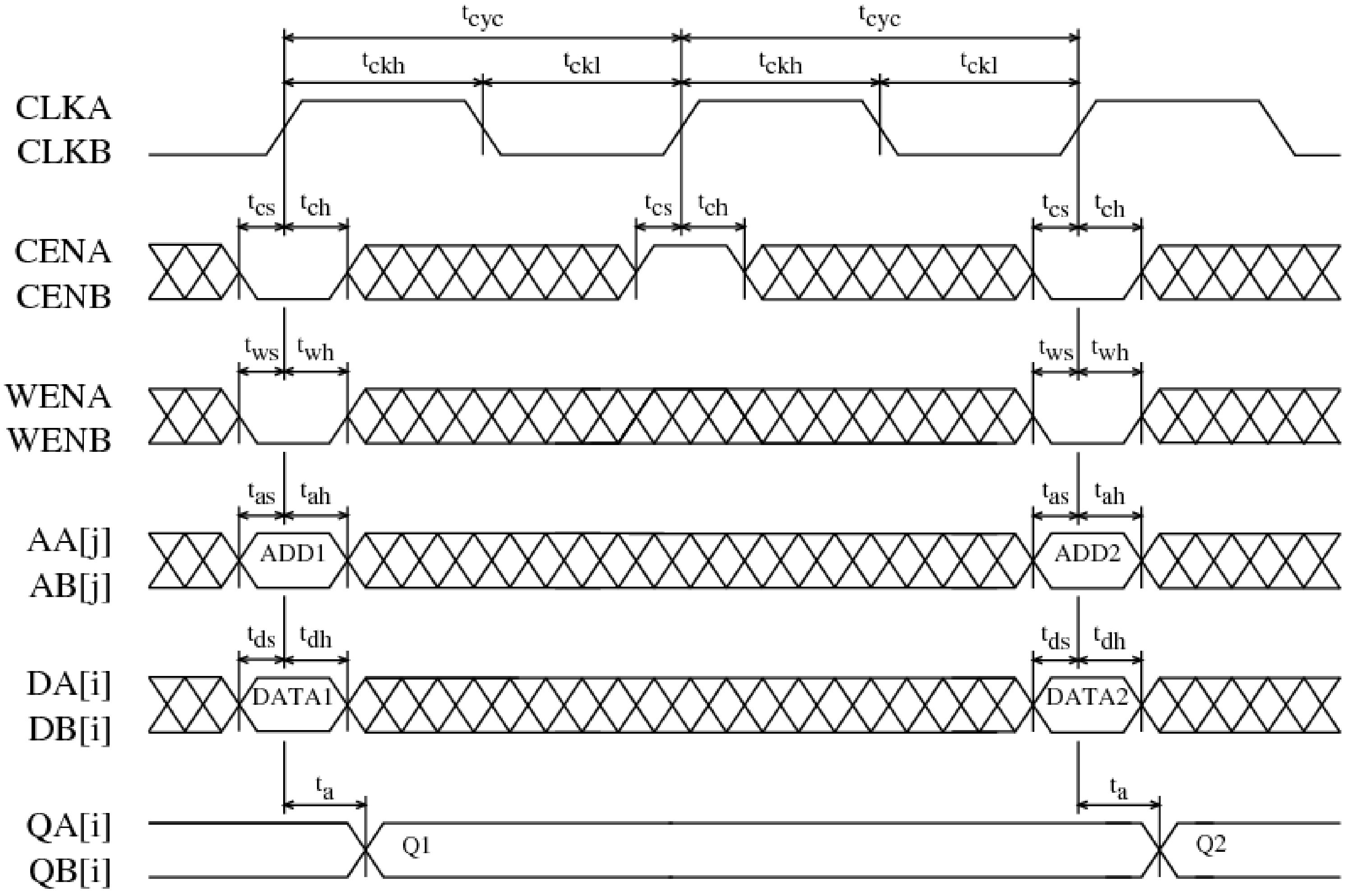


Read Cycle Timing (Dual Port)

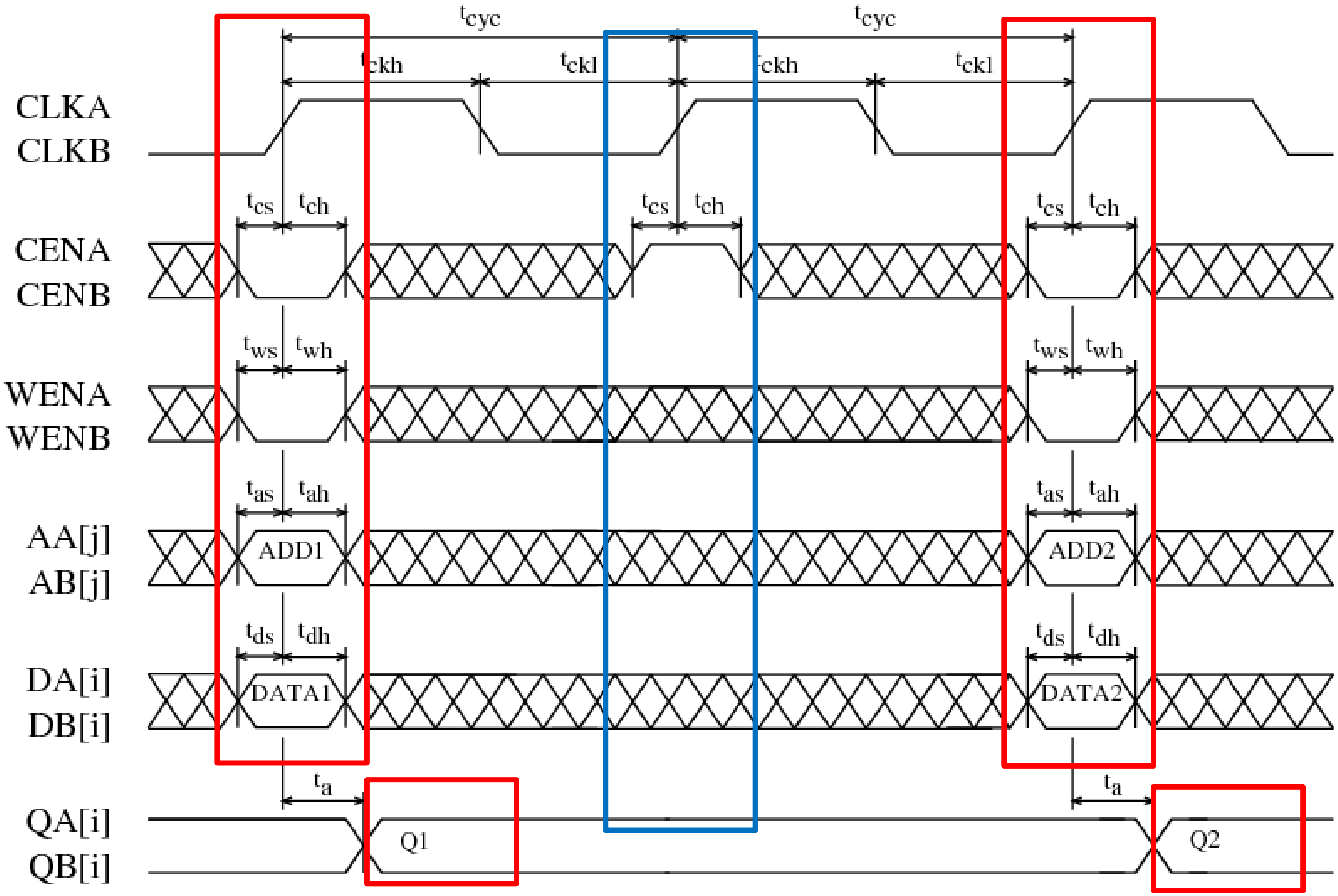


CENA、CENB	1’b0 (enable)	1’b1 (disable)	1’b0 (enable)
WENA、WENB	1’b1 (read)	X	1’b1 (read)

Write Cycle Timing (Dual Port)



Write Cycle Timing (Dual Port)



CENA、CENB	1’b0 (enable)	1’b1 (disable)	1’b0 (enable)
WENA、WENB	1’b0 (write)	X	1’b1 (write)

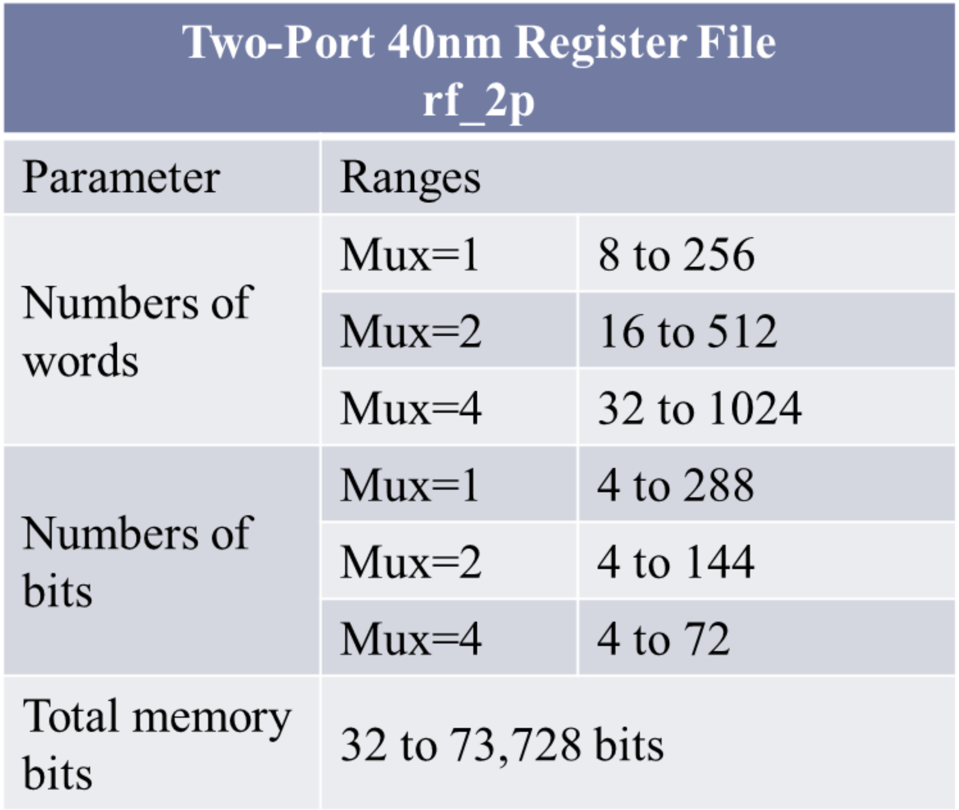
ARM memory parameter（1/4）

- Register File

High Speed Single-Port 40nm Register File rf_sp_hsd			High Density Single-Port 40nm Register File rf_sp_hde		
Parameter	Ranges		Parameter	Ranges	
Numbers of words	Mux=2	8 to 256	Numbers of words	Mux=2	16 to 512
	Mux=4	16 to 512		Mux=4	32 to 1024
	Mux=8	32 to 1024		Mux=8	32 to 2048
Numbers of bits	Mux=2	4 to 144	Numbers of bits	Mux=2	4 to 144
	Mux=4	4 to 72		Mux=4	4 to 144
	Mux=8	4 to 36		Mux=8	4 to 72
Total memory bits	32 to 36,384 bits		Total memory bits	64 to 147,456 bits	

ARM memory parameter（2/4）

- Register File



Two-Port 40nm Register File
rf_2p

Parameter	Ranges	
Numbers of words	Mux=1	8 to 256
	Mux=2	16 to 512
	Mux=4	32 to 1024
Numbers of bits	Mux=1	4 to 288
	Mux=2	4 to 144
	Mux=4	4 to 72
Total memory bits	32 to 73,728 bits	

ARM memory parameter（3/4）

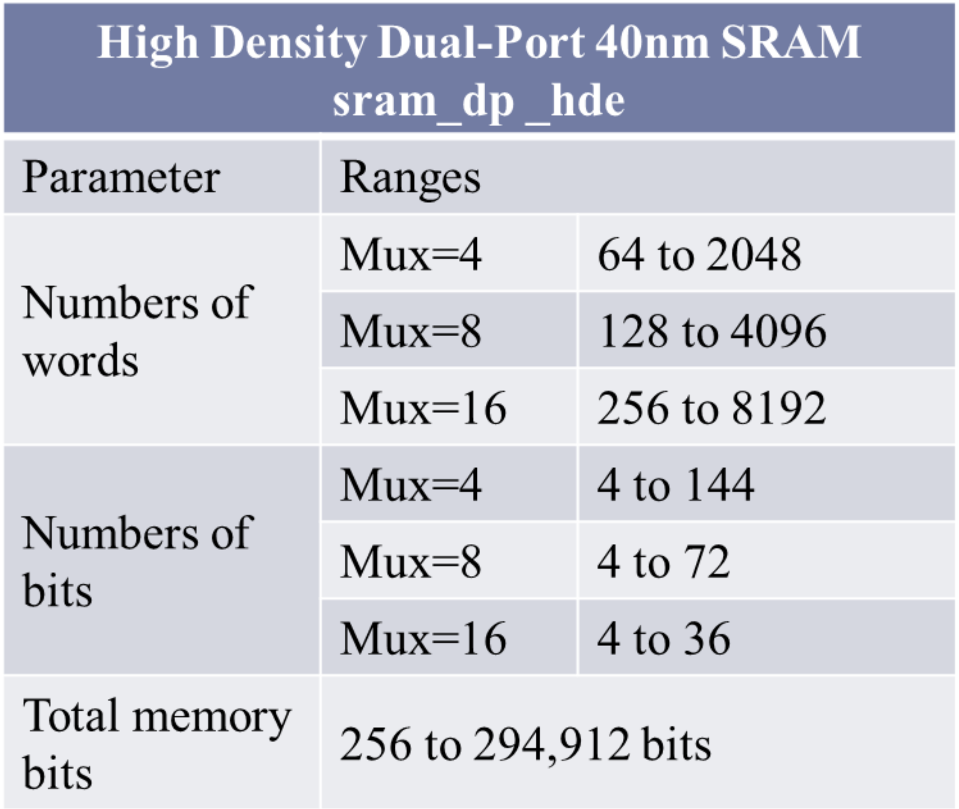
- SRAM

High Density Single-Port 40nm SRAM sram_sp_hde		
Parameter	Ranges	
Numbers of words	Mux=8	256 to 4096
	Mux=16	512 to 8192
	Mux=32	1024 to 16384
Numbers of bits	Mux=8	4 to 144
	Mux=16	4 to 72
	Mux=32	4 to 36
Total memory bits	1024 to 589,824 bits	

High Speed Single-Port 40nm SRAM sram_sp_hsc		
Parameter	Ranges	
Numbers of words	Mux=8	256 to 4096
	Mux=16	512 to 8192
	Mux=32	1024 to 16384
Numbers of bits	Mux=8	4 to 144
	Mux=16	4 to 72
	Mux=32	4 to 36
Total memory bits	512 to 589,824 bits	

ARM memory parameter（4/4）

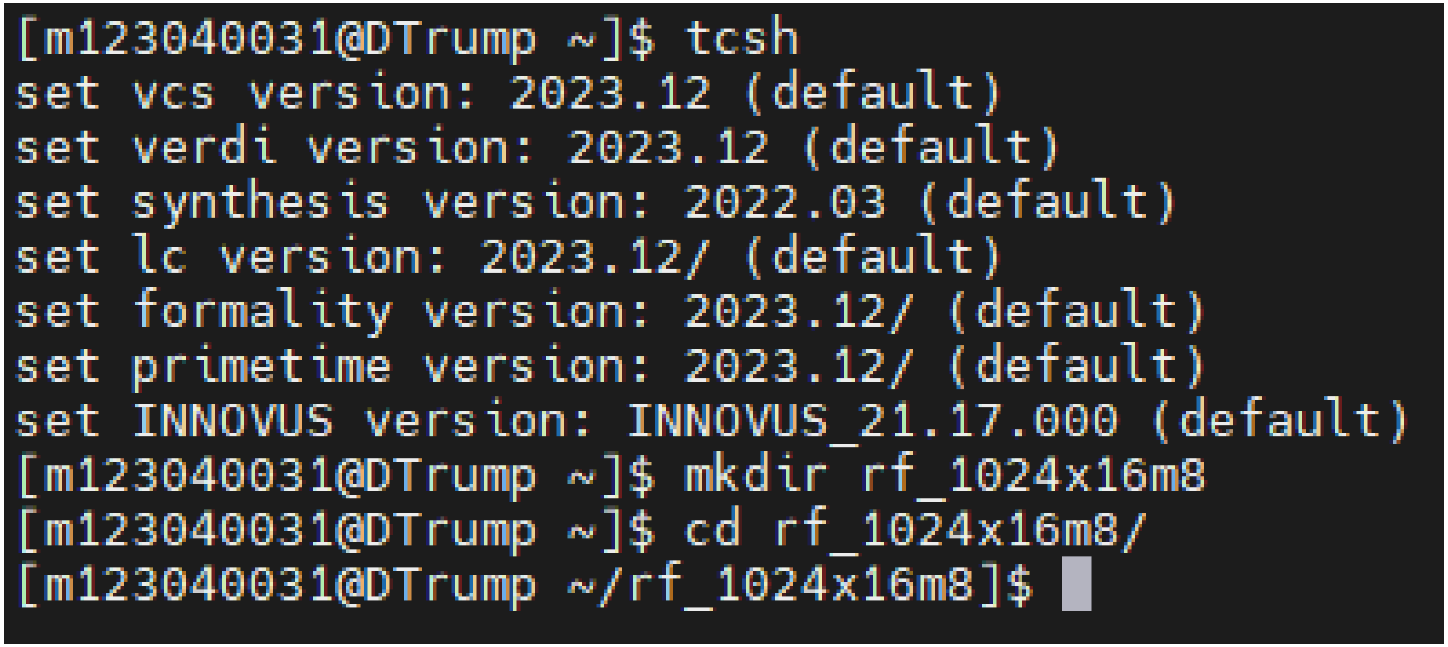
- SRAM



High Density Dual-Port 40nm SRAM sram_dp _hde		
Parameter	Ranges	
Numbers of words	Mux=4	64 to 2048
	Mux=8	128 to 4096
	Mux=16	256 to 8192
Numbers of bits	Mux=4	4 to 144
	Mux=8	4 to 72
	Mux=16	4 to 36
Total memory bits	256 to 294,912 bits	

Memory Compiler Flow（1/10）

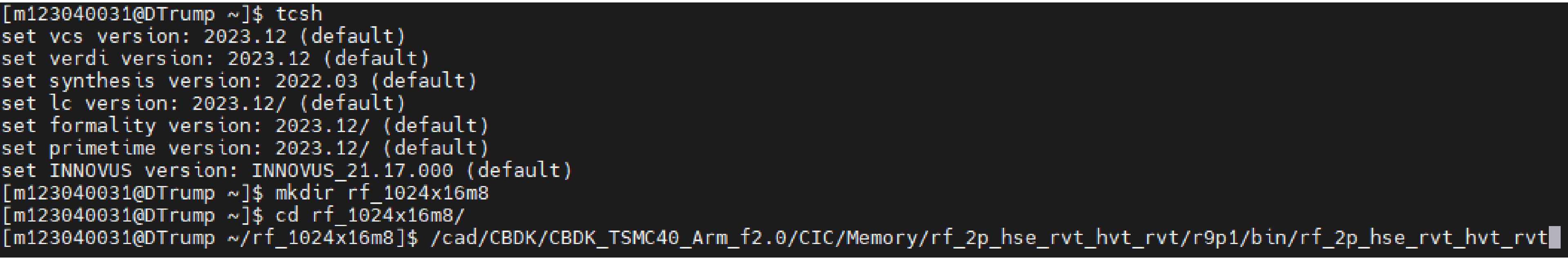
- Step1.開啟介面
  - 連線到Linux作業系統的工作站並進入到自己創的資料夾目錄
  - 此範例創建的資料夾為rf_1024x16m8，創好後cd進入



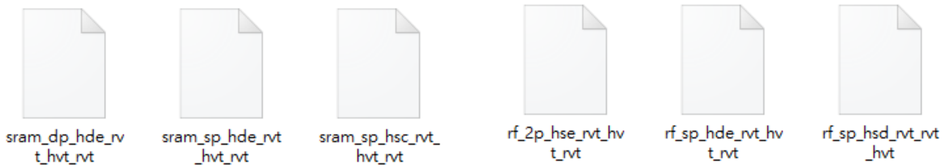
```
[m123040031@DTrump ~]$ tcsh
set vcs version: 2023.12 (default)
set verdi version: 2023.12 (default)
set synthesis version: 2022.03 (default)
set lc version: 2023.12/ (default)
set formality version: 2023.12/ (default)
set primetime version: 2023.12/ (default)
set INNOVUS version: INNOVUS_21.17.000 (default)
[m123040031@DTrump ~]$ mkdir rf_1024x16m8
[m123040031@DTrump ~]$ cd rf_1024x16m8/
[m123040031@DTrump ~/rf_1024x16m8]$ 
```

Memory Compiler Flow（2/10）

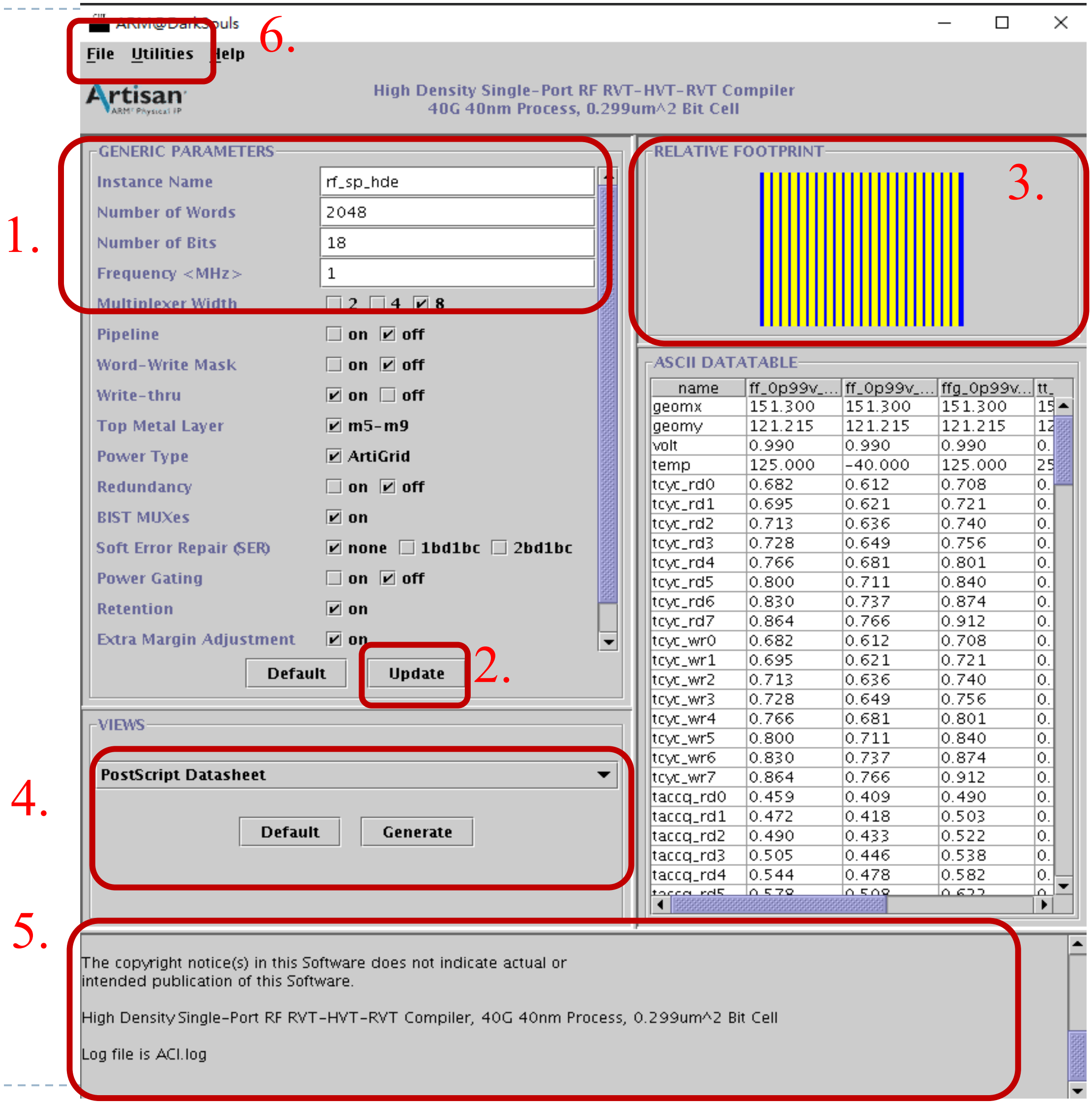
- Step2.開啟ARM Memory Compiler
  - 進到自己創的資料夾後輸入下列指令來開啟ARM Memory Compiler
  - /cad/CBDK/CBDK_TSMC40_Arm_f2.0/CIC/Memory/rf_sp_hde_rvt_hvt_rvt/r8p2/bin/rf_sp_hde_rvt_hvt_rvt(這裡以register file_single port_high density舉例)
  - 若需要產生其他的Memory可以到/cad/CBDK/CBDK_TSMC40_Arm_f2.0/CIC/Memory內找相對應的檔案 (完整路徑在備忘稿中)



```
[m123040031@DTrump ~]$ tcsh
set vcs version: 2023.12 (default)
set verdi version: 2023.12 (default)
set synthesis version: 2022.03 (default)
set lc version: 2023.12/ (default)
set formality version: 2023.12/ (default)
set primetime version: 2023.12/ (default)
set INNOVUS version: INNOVUS_21.17.000 (default)
[m123040031@DTrump ~]$ mkdir rf_1024x16m8
[m123040031@DTrump ~]$ cd rf_1024x16m8/
[m123040031@DTrump ~/rf_1024x16m8]$ /cad/CBDK/CBDK_TSMC40_Arm_f2.0/CIC/Memory/rf_2p_hse_rvt_hvt_rvt/r9p1/bin/rf_2p_hse_rvt_hvt_rvt
```



Memory Compiler Flow（3/10）



High Density Single-Port RF RVT-HVT-RVT Compiler
40G 40nm Process, 0.299um^2 Bit Cell

GENERIC PARAMETERS

Instance Name	rf_sp_hde
Number of Words	2048
Number of Bits	18
Frequency <MHz>	1
Multiplexer Width	□ 2 □ 4 ☑ 8
Pipeline	□ on ☑ off
Word-Write Mask	□ on ☑ off
Write-thru	☑ on □ off
Top Metal Layer	☑ m5-m9
Power Type	☑ ArtiGrid
Redundancy	□ on ☑ off
BIST MUXes	☑ on
Soft Error Repair (SER)	☑ none □ 1bd1bc □ 2bd1bc
Power Gating	□ on ☑ off
Retention	☑ on
Extra Margin Adjustment	☑ on

PostScript Datasheet

ASCII DATATABLE

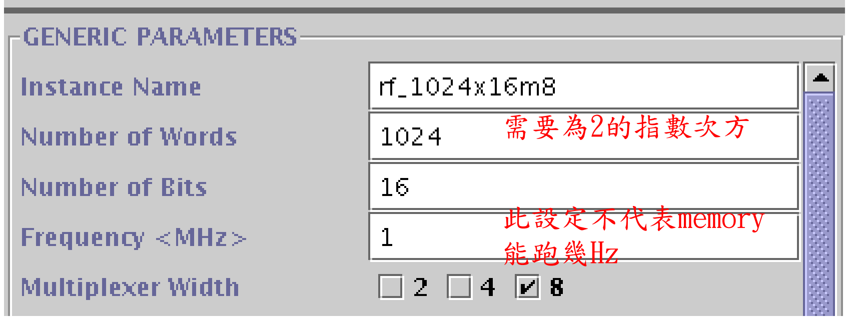
name	ff_0p99v_...	ff_0p99v_...	ffg_0p99v...	tt_
geomx	151.300	151.300	151.300	15
geomy	121.215	121.215	121.215	12
volt	0.990	0.990	0.990	0.
temp	125.000	-40.000	125.000	25
tcyc_rd0	0.682	0.612	0.708	0.
tcyc_rd1	0.695	0.621	0.721	0.
tcyc_rd2	0.713	0.636	0.740	0.
tcyc_rd3	0.728	0.649	0.756	0.
tcyc_rd4	0.766	0.681	0.801	0.
tcyc_rd5	0.800	0.711	0.840	0.
tcyc_rd6	0.830	0.737	0.874	0.
tcyc_rd7	0.864	0.766	0.912	0.
tcyc_wr0	0.682	0.612	0.708	0.
tcyc_wr1	0.695	0.621	0.721	0.
tcyc_wr2	0.713	0.636	0.740	0.
tcyc_wr3	0.728	0.649	0.756	0.
tcyc_wr4	0.766	0.681	0.801	0.
tcyc_wr5	0.800	0.711	0.840	0.
tcyc_wr6	0.830	0.737	0.874	0.
tcyc_wr7	0.864	0.766	0.912	0.
taccq_rd0	0.459	0.409	0.490	0.
taccq_rd1	0.472	0.418	0.503	0.
taccq_rd2	0.490	0.433	0.522	0.
taccq_rd3	0.505	0.446	0.538	0.
taccq_rd4	0.544	0.478	0.582	0.
taccq_rd5	0.578	0.508	0.622	0.

```
The copyright notice(s) in this Software does not indicate actual or
intended publication of this Software.

High Density Single-Port RF RVT-HVT-RVT Compiler, 40G 40nm Process, 0.299um^2 Bit Cell

Log file is ACI.log
```

Memory Compiler Flow（4/10）

- Step3.產生Memory Verilog Model (for verilog simulation)
  - EX：rf_1024x16m8 大小：1024x16 mux:8(名稱可以自訂)

1.輸入Memory名稱、大小與Mux寬度



GENERIC PARAMETERS

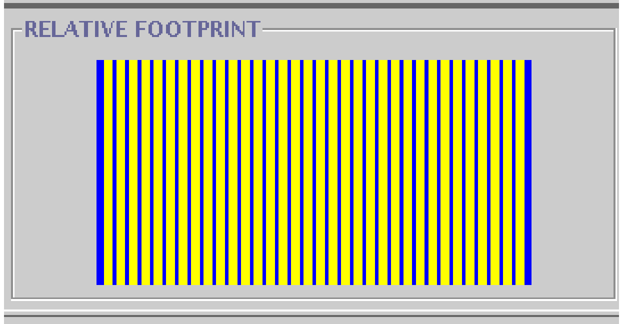
Instance Name	rf_1024x16m8
Number of Words	1024
Number of Bits	16
Frequency <MHz>	1
Multiplexer Width	□ 2 □ 4 ☑ 8

需要為2的指數次方

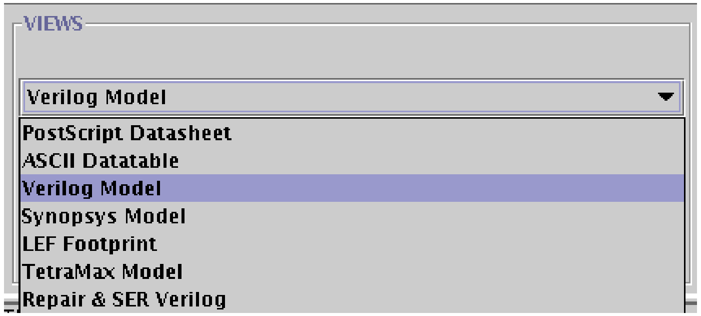
此設定不代表memory能跑幾Hz

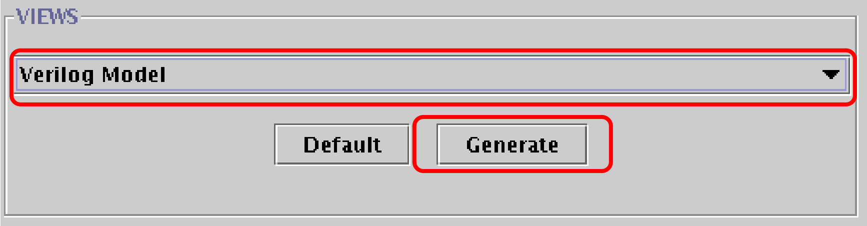
2.按下update按鈕

3.檢查是否有Memory示意圖出現

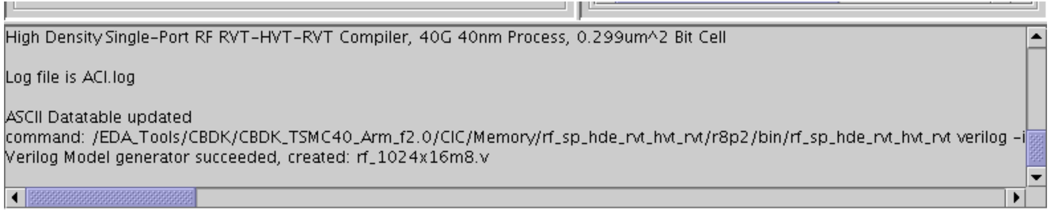


4.VIEWS選擇Verilog Model，並按下Generate按鈕





5.檢查是否有成功產生，rf_1024x16m8.v檔案

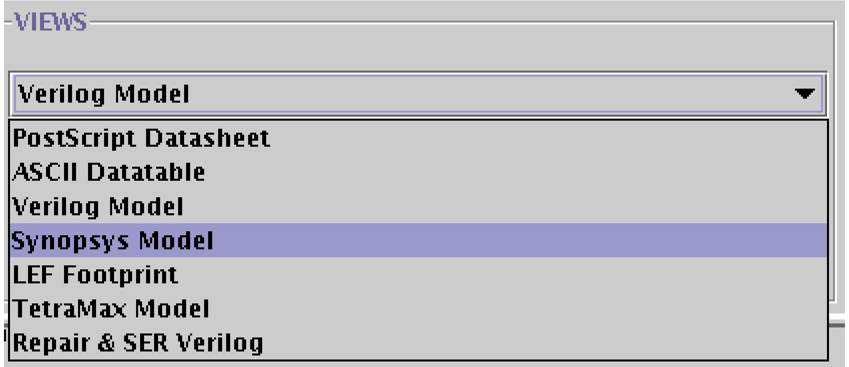


```
High Density Single-Port RF RVT-HVT-RVT Compiler, 40G 40nm Process, 0.299um^2 Bit Cell

Log file is ACI.log

ASCII Datatable updated
command: /EDA_Tools/CBDK/CBDK_TSMC40_Arm_f2.0/CIC/Memory/rf_sp_hde_rvt_hvt_rvt/r8p2/bin/rf_sp_hde_rvt_hvt_rvt verilog -i
Verilog Model generator succeeded, created: rf_1024x16m8.v
```

Memory Compiler Flow（5/10）

- Step4.產生Synopsys Model (For Design Complier)

1.VIEWS選擇Synopsys Model

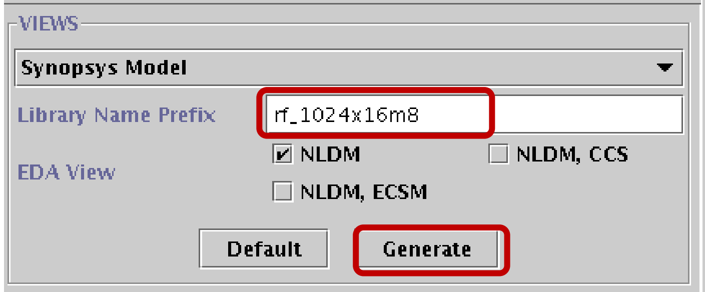


VIEWS

Verilog Model ▼

PostScript Datasheet
ASCII Datatable
Verilog Model
Synopsys Model
LEF Footprint
TetraMax Model
Repair & SER Verilog

2.Library Name輸入Memory名稱(可自訂)



VIEWS

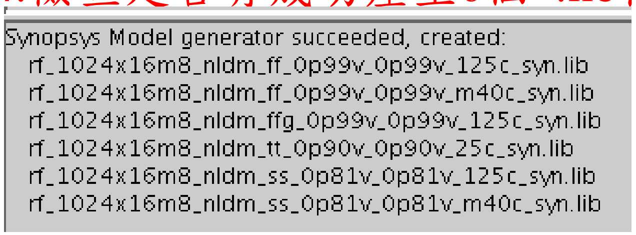
Synopsys Model ▼

Library Name Prefix: rf_1024x16m8

EDA View: ☑ NLDM □ NLDM, CCS □ NLDM, ECSM

3.按下Generate按鈕(會花點時間)

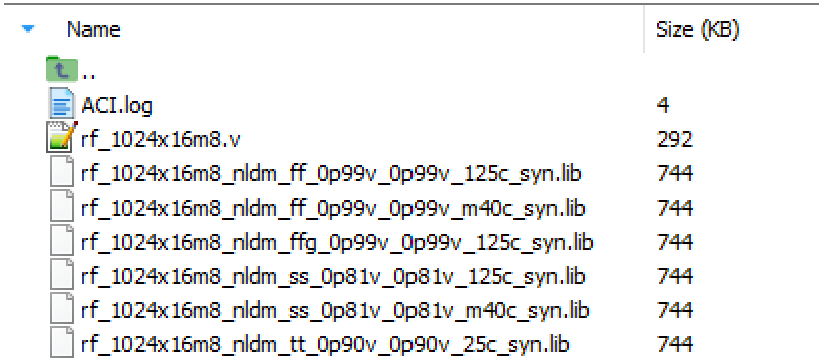
4.檢查是否有成功產生6個*.lib檔案



```
Synopsys Model generator succeeded, created:
   rf_1024x16m8_nldm_ff_0p99v_0p99v_125c_syn.lib
   rf_1024x16m8_nldm_ff_0p99v_0p99v_m40c_syn.lib
   rf_1024x16m8_nldm_ffg_0p99v_0p99v_125c_syn.lib
   rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.lib
   rf_1024x16m8_nldm_ss_0p81v_0p81v_125c_syn.lib
   rf_1024x16m8_nldm_ss_0p81v_0p81v_m40c_syn.lib
```

Memory Generator會針對不同的操作狀況
（ fast、slow、typical)_ voltage _ temperature
分別產生不同的*.lib檔案
Ex: ff_0pxxv_xxc //fast_voltage xxV & temperature = xx度C

5.做到這步驟會有以下這些檔案

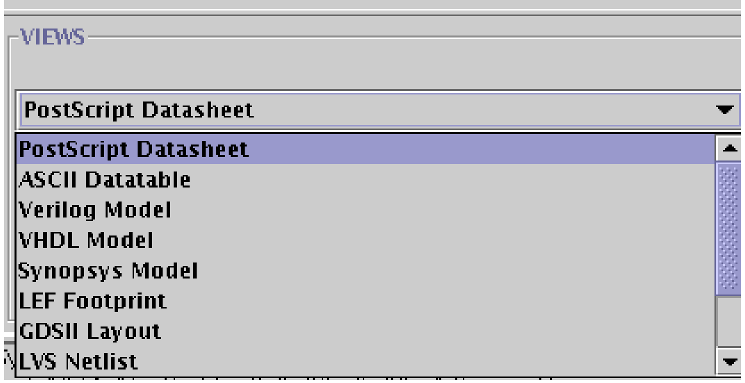


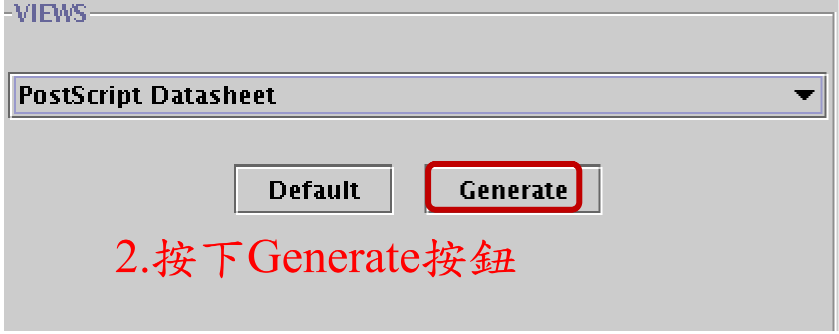
Name	Size (KB)
..	
ACI.log	4
rf_1024x16m8.v	292
rf_1024x16m8_nldm_ff_0p99v_0p99v_125c_syn.lib	744
rf_1024x16m8_nldm_ff_0p99v_0p99v_m40c_syn.lib	744
rf_1024x16m8_nldm_ffg_0p99v_0p99v_125c_syn.lib	744
rf_1024x16m8_nldm_ss_0p81v_0p81v_125c_syn.lib	744
rf_1024x16m8_nldm_ss_0p81v_0p81v_m40c_syn.lib	744
rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.lib	744

Memory Compiler Flow（6/10）

- Step5.產生PS檔(可做或可不做)
- PS檔是為了看產生好的memory的Datasheet(伺服器無ps2pdf的指令)
- (之後在terminal 打ps2pdf rf_1024x16m8_tt_0p90v_0p90v_25c.ps rf_1024x16m8_tt.pdf)

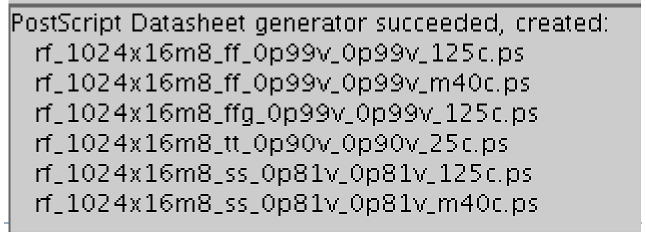
1.VIEWS選擇PostScript Datasheet





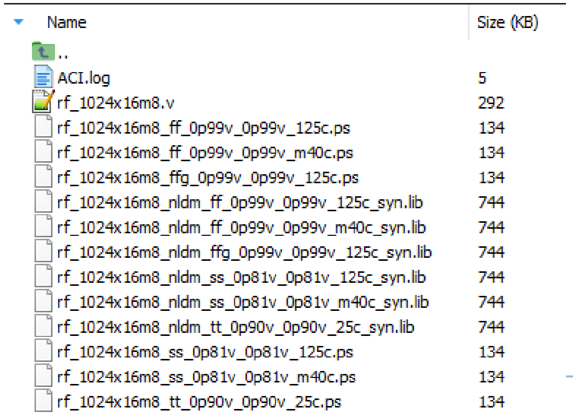
2.按下Generate按鈕

3.檢查是否有成功產生不同corner下的ps檔



```
PostScript Datasheet generator succeeded, created:
  rf_1024x16m8_ff_0p99v_0p99v_125c.ps
  rf_1024x16m8_ff_0p99v_0p99v_m40c.ps
  rf_1024x16m8_ffg_0p99v_0p99v_125c.ps
  rf_1024x16m8_tt_0p90v_0p90v_25c.ps
  rf_1024x16m8_ss_0p81v_0p81v_125c.ps
  rf_1024x16m8_ss_0p81v_0p81v_m40c.ps
```

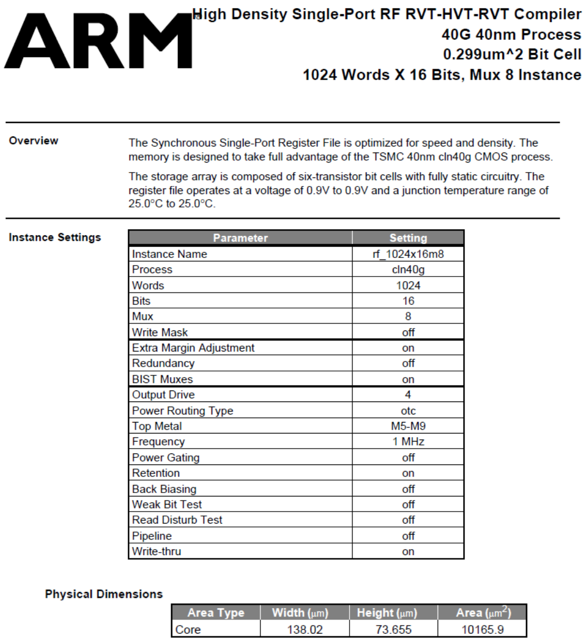
4.做到這步驟會有以下這些檔案



Name	Size (KB)
..	
ACI.log	5
rf_1024x16m8.v	292
rf_1024x16m8_ff_0p99v_0p99v_125c.ps	134
rf_1024x16m8_ff_0p99v_0p99v_m40c.ps	134
rf_1024x16m8_ffg_0p99v_0p99v_125c.ps	134
rf_1024x16m8_nldm_ff_0p99v_0p99v_125c_syn.lib	744
rf_1024x16m8_nldm_ff_0p99v_0p99v_m40c_syn.lib	744
rf_1024x16m8_nldm_ffg_0p99v_0p99v_125c_syn.lib	744
rf_1024x16m8_nldm_ss_0p81v_0p81v_125c_syn.lib	744
rf_1024x16m8_nldm_ss_0p81v_0p81v_m40c_syn.lib	744
rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.lib	744
rf_1024x16m8_ss_0p81v_0p81v_125c.ps	134
rf_1024x16m8_ss_0p81v_0p81v_m40c.ps	134
rf_1024x16m8_tt_0p90v_0p90v_25c.ps	134

Memory Compiler Flow（7/10）

- rf_1024x16m8_tt.pdf内容



ARM

High Density Single-Port RF RVT-HVT-RVT Compiler
40G 40nm Process
0.299um^2 Bit Cell
1024 Words X 16 Bits, Mux 8 Instance

Overview

The Synchronous Single-Port Register File is optimized for speed and density. The memory is designed to take full advantage of the TSMC 40nm cln40g CMOS process.

The storage array is composed of six-transistor bit cells with fully static circuitry. The register file operates at a voltage of 0.9V to 0.9V and a junction temperature range of 25.0°C to 25.0°C.

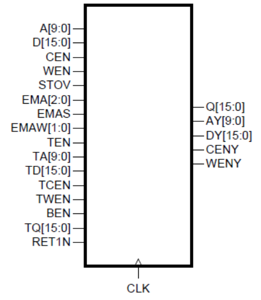
Instance Settings

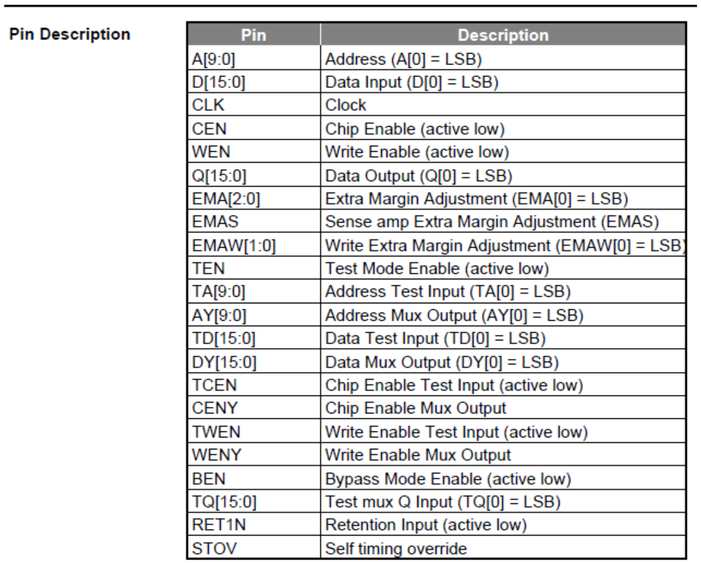
Parameter	Setting
Instance Name	rf_1024x16m8
Process	cln40g
Words	1024
Bits	16
Mux	8
Write Mask	off
Extra Margin Adjustment	on
Redundancy	off
BIST Muxes	on
Output Drive	4
Power Routing Type	otc
Top Metal	M5-M9
Frequency	1 MHz
Power Gating	off
Retention	on
Back Biasing	off
Weak Bit Test	off
Read Disturb Test	off
Pipeline	off
Write-thru	on

Physical Dimensions

Area Type	Width (μm)	Height (μm)	Area (μm^2)
Core	138.02	73.655	10165.9

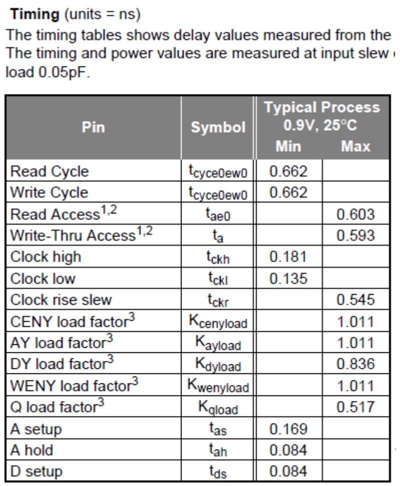
Symbol





Pin Description

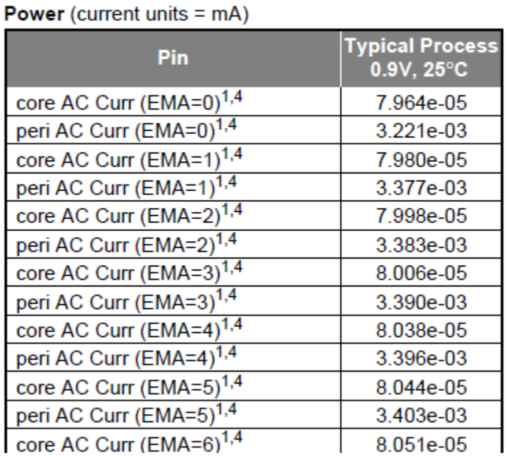
Pin	Description
A[9:0]	Address (A[0] = LSB)
D[15:0]	Data Input (D[0] = LSB)
CLK	Clock
CEN	Chip Enable (active low)
WEN	Write Enable (active low)
Q[15:0]	Data Output (Q[0] = LSB)
EMA[2:0]	Extra Margin Adjustment (EMA[0] = LSB)
EMAS	Sense amp Extra Margin Adjustment (EMAS)
EMAW[1:0]	Write Extra Margin Adjustment (EMAW[0] = LSB)
TEN	Test Mode Enable (active low)
TA[9:0]	Address Test Input (TA[0] = LSB)
AY[9:0]	Address Mux Output (AY[0] = LSB)
TD[15:0]	Data Test Input (TD[0] = LSB)
DY[15:0]	Data Mux Output (DY[0] = LSB)
TCEN	Chip Enable Test Input (active low)
CENY	Chip Enable Mux Output
TWEN	Write Enable Test Input (active low)
WENY	Write Enable Mux Output
BEN	Bypass Mode Enable (active low)
TQ[15:0]	Test mux Q Input (TQ[0] = LSB)
RET1N	Retention Input (active low)
STOV	Self timing override



Timing (units = ns)

The timing tables shows delay values measured from the
The timing and power values are measured at input slew
load 0.05pF.

Pin	Symbol	Typical Process 0.9V, 25°C	
		Min	Max
Read Cycle	$t_{cyce0ew0}$	0.662	
Write Cycle	$t_{cyce0ew0}$	0.662	
Read Access[1,2]	t_{ae0}		0.603
Write-Thru Access[1,2]	t_a		0.593
Clock high	t_{ckh}	0.181	
Clock low	t_{ckl}	0.135	
Clock rise slew	t_{ckr}		0.545
CENY load factor[3]	$K_{cenyload}$		1.011
AY load factor[3]	K_{ayload}		1.011
DY load factor[3]	K_{dyload}		0.836
WENY load factor[3]	$K_{wenyload}$		1.011
Q load factor[3]	K_{qload}		0.517
A setup	t_{as}	0.169	
A hold	t_{ah}	0.084	
D setup	t_{ds}	0.084	



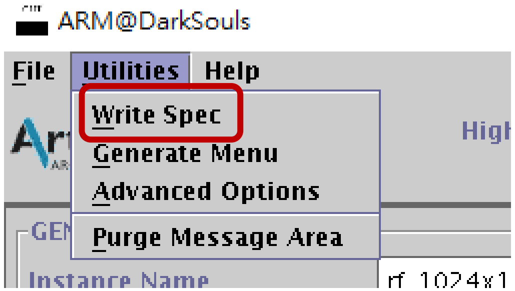
Power (current units = mA)

Pin	Typical Process 0.9V, 25°C
core AC Curr (EMA=0)[1,4]	7.964e-05
peri AC Curr (EMA=0)[1,4]	3.221e-03
core AC Curr (EMA=1)[1,4]	7.980e-05
peri AC Curr (EMA=1)[1,4]	3.377e-03
core AC Curr (EMA=2)[1,4]	7.998e-05
peri AC Curr (EMA=2)[1,4]	3.383e-03
core AC Curr (EMA=3)[1,4]	8.006e-05
peri AC Curr (EMA=3)[1,4]	3.390e-03
core AC Curr (EMA=4)[1,4]	8.038e-05
peri AC Curr (EMA=4)[1,4]	3.396e-03
core AC Curr (EMA=5)[1,4]	8.044e-05
peri AC Curr (EMA=5)[1,4]	3.403e-03
core AC Curr (EMA=6)[1,4]	8.051e-05

Memory Compiler Flow（8/10）

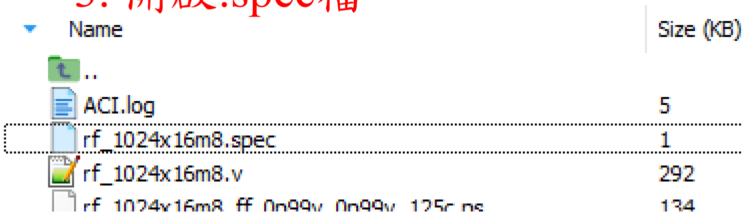
- Step6.產生Spec(可做或可不做)
- 產生出spec檔，內容為在memory compiler所設定的參數與名稱

1.點選左上方的Utilities



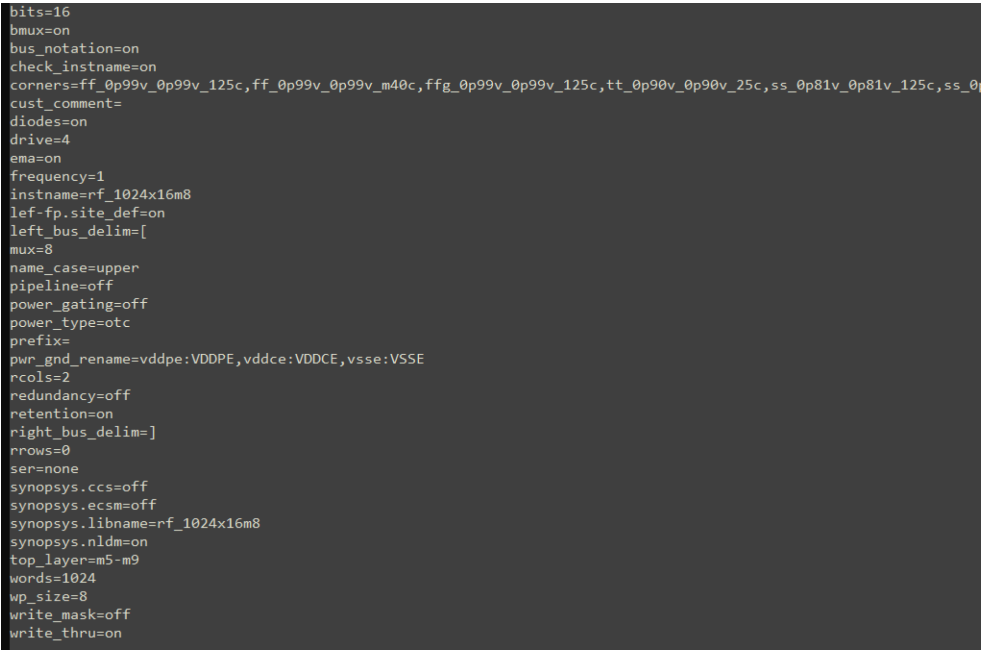
2. 點Write Spec

3. 開啟.spec檔



Name	Size (KB)
..	
ACI.log	5
rf_1024x16m8.spec	1
rf_1024x16m8.v	292
rf_1024x16m8_ff_0p99v_0p99v_125c.ps	134

4. 可看到在memory compiler所設定的參數與名稱等

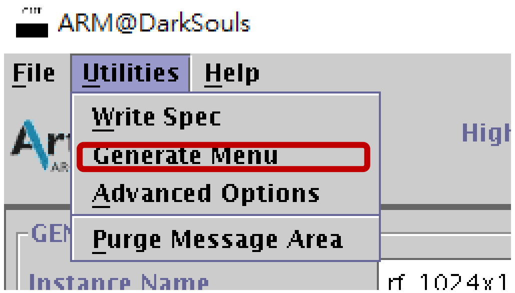


```
bits=16
bmux=on
bus_notation=on
check_instname=on
corners=ff_0p99v_0p99v_125c,ff_0p99v_0p99v_m40c,ffg_0p99v_0p99v_125c,tt_0p90v_0p90v_25c,ss_0p81v_0p81v_125c,ss_0
cust_comment=
diodes=on
drive=4
ema=on
frequency=1
instname=rf_1024x16m8
lef-fp.site_def=on
left_bus_delim=[
mux=8
name_case=upper
pipeline=off
power_gating=off
power_type=otc
prefix=
pwr_gnd_rename=vddpe:VDDPE,vddce:VDDCE,vsse:VSSE
rcols=2
redundancy=off
retention=on
right_bus_delim=]
rrows=0
ser=none
synopsys.ccs=off
synopsys.ecsm=off
synopsys.libname=rf_1024x16m8
synopsys.nldm=on
top_layer=m5-m9
words=1024
wp_size=8
write_mask=off
write_thru=on
```

Memory Compiler Flow（9/10）

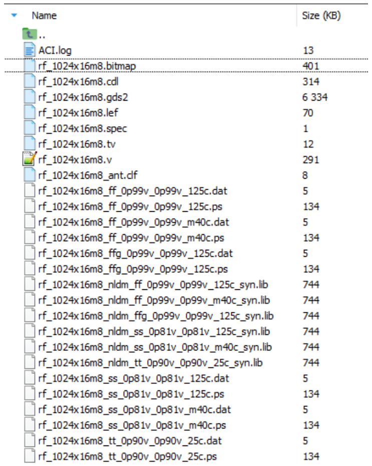
- Step7.產生所有檔案(可做或可不做)

1.點選左上方的Utilities



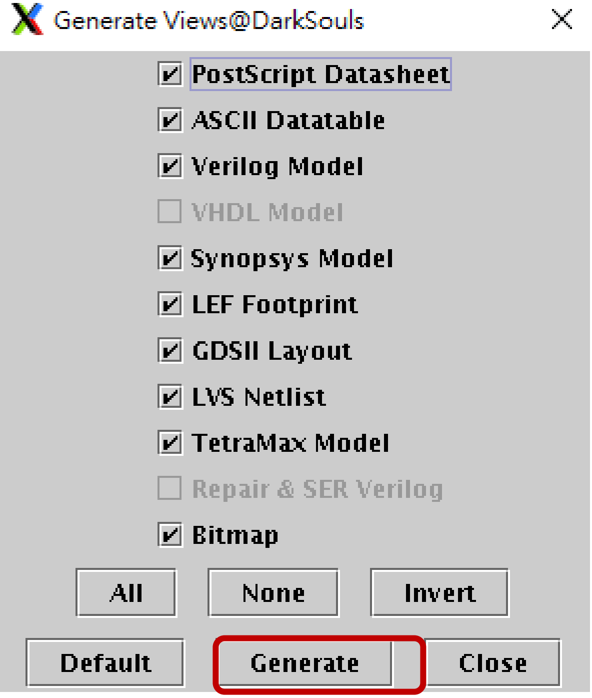
2. 點Generate Menu

5. 產生結果



Name	Size (KB)
..	
ACI.log	13
rf_1024x16m8.bitmap	401
rf_1024x16m8.cdl	314
rf_1024x16m8.gds2	6 334
rf_1024x16m8.lef	70
rf_1024x16m8.spec	1
rf_1024x16m8.tv	12
rf_1024x16m8.v	291
rf_1024x16m8_ant.clf	8
rf_1024x16m8_ff_0p99v_0p99v_125c.dat	5
rf_1024x16m8_ff_0p99v_0p99v_125c.ps	134
rf_1024x16m8_ff_0p99v_0p99v_m40c.dat	5
rf_1024x16m8_ff_0p99v_0p99v_m40c.ps	134
rf_1024x16m8_ffg_0p99v_0p99v_125c.dat	5
rf_1024x16m8_ffg_0p99v_0p99v_125c.ps	134
rf_1024x16m8_nldm_ff_0p99v_0p99v_125c_syn.lib	744
rf_1024x16m8_nldm_ff_0p99v_0p99v_m40c_syn.lib	744
rf_1024x16m8_nldm_ffg_0p99v_0p99v_125c_syn.lib	744
rf_1024x16m8_nldm_ss_0p81v_0p81v_125c_syn.lib	744
rf_1024x16m8_nldm_ss_0p81v_0p81v_m40c_syn.lib	744
rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.lib	744
rf_1024x16m8_ss_0p81v_0p81v_125c.dat	5
rf_1024x16m8_ss_0p81v_0p81v_125c.ps	134
rf_1024x16m8_ss_0p81v_0p81v_m40c.dat	5
rf_1024x16m8_ss_0p81v_0p81v_m40c.ps	134
rf_1024x16m8_tt_0p90v_0p90v_25c.dat	5
rf_1024x16m8_tt_0p90v_0p90v_25c.ps	134

3. 點Generate，即可得到勾選使用的檔案



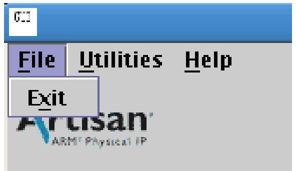
Generate Views@DarkSouls

☑ PostScript Datasheet
☑ ASCII Datatable
☑ Verilog Model
☐ VHDL Model
☑ Synopsys Model
☑ LEF Footprint
☑ GDSII Layout
☑ LVS Netlist
☑ TetraMax Model
☐ Repair & SER Verilog
☑ Bitmap

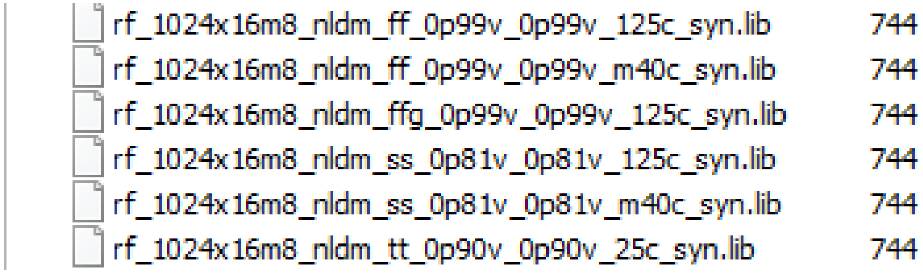
4. 注意Synopsys Model若沒有在step4設定lib name則會用 “USRLIB”為預設名字，所以step4的名稱要設定。

Memory Compiler Flow（10/10）

- Step8. 關掉Memory Compiler
  - File→Exit



- Step9. 將Memory Compiler產生的檔案放至目錄下
  - 通常我們需要*.v檔及*.lib檔
  - 由於Synopsys Design Compiler無法直接使用產生的*.lib檔案
  - 因此要先將*.lib檔案轉為Design Compiler可使用的*.db檔案



```
rf_1024x16m8_nldm_ff_0p99v_0p99v_125c_syn.lib     744
rf_1024x16m8_nldm_ff_0p99v_0p99v_m40c_syn.lib     744
rf_1024x16m8_nldm_ffg_0p99v_0p99v_125c_syn.lib    744
rf_1024x16m8_nldm_ss_0p81v_0p81v_125c_syn.lib     744
rf_1024x16m8_nldm_ss_0p81v_0p81v_m40c_syn.lib     744
rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.lib      744
```

Memory lib to db Flow（1/3）

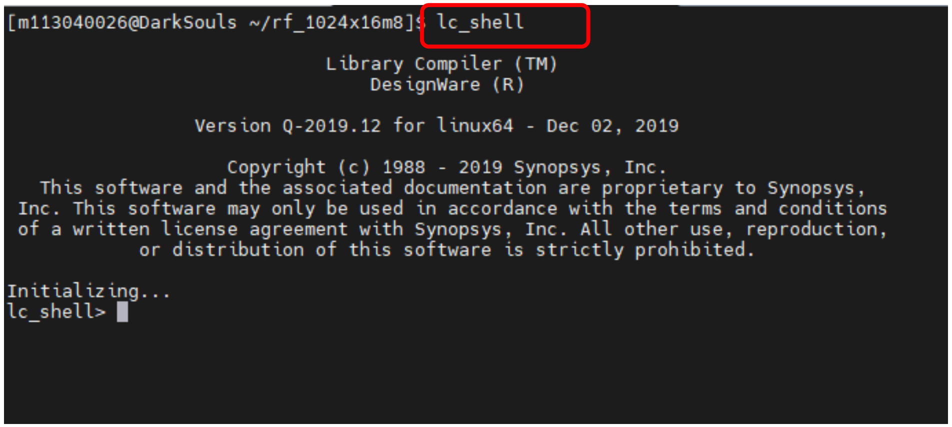
- Step1. 將*.lib檔compile成為*.db檔
  - 開啟Terminal輸入下列指令(以tt 0.9v 25c 作為範例)
  - lc_shell
  - read_lib rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.lib
  - write_lib rf_1024x16m8_nldm_tt_0p90v_0p90v_25c -output rf_1024x16m8_nldm_tt_0p90v_0p90v_25c.db
  - 這邊的Library name input terminal 輸出的名字 不要複製檔案名
  - 會順利產生rf_1024x16m8_nldm_tt_0p90v_0p90v_25c.db檔
  - exit

- 若需要產生多個db檔可以寫tcl腳本 並用 lc_shell –f xxx.tcl使用

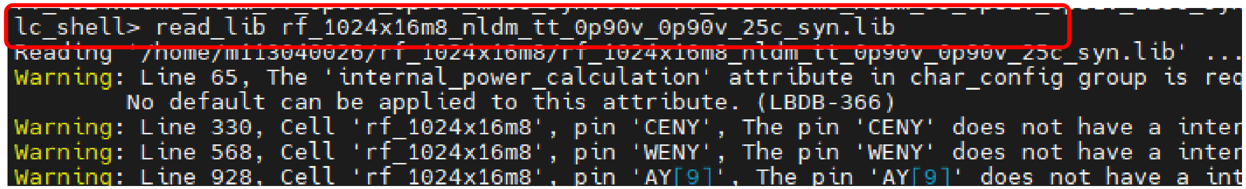
★★★ 複製: Ctrl + C

貼上: 滑鼠右鍵點一下

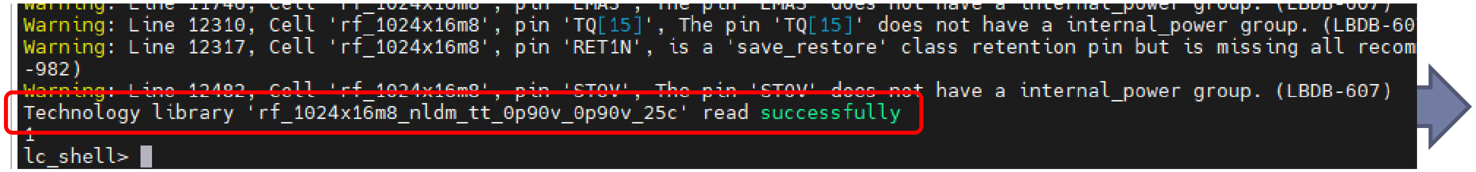
Memory lib to db Flow（2/3）



```
[m113040026@DarkSouls ~/rf_1024x16m8]$ lc_shell

                           Library Compiler (TM)
                              DesignWare (R)

                   Version Q-2019.12 for linux64 - Dec 02, 2019

                     Copyright (c) 1988 - 2019 Synopsys, Inc.
   This software and the associated documentation are proprietary to Synopsys,
 Inc. This software may only be used in accordance with the terms and conditions
 of a written license agreement with Synopsys, Inc. All other use, reproduction,
            or distribution of this software is strictly prohibited.

Initializing...
lc_shell> 
```

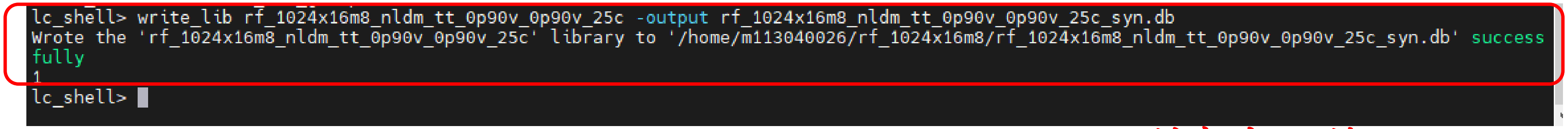


```
lc_shell> read_lib rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.lib
Reading '/home/m113040026/rf_1024x16m8/rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.lib' ...
Warning: Line 65, The 'internal_power_calculation' attribute in char_config group is req
         No default can be applied to this attribute. (LBDB-366)
Warning: Line 330, Cell 'rf_1024x16m8', pin 'CENY', The pin 'CENY' does not have a inter
Warning: Line 568, Cell 'rf_1024x16m8', pin 'WENY', The pin 'WENY' does not have a inter
Warning: Line 928, Cell 'rf_1024x16m8', pin 'AY[9]', The pin 'AY[9]' does not have a int
```



```
Warning: Line 11740, Cell 'rf_1024x16m8', pin 'EMAS', The pin 'EMAS' does not have a internal_power group. (LBDB-607)
Warning: Line 12310, Cell 'rf_1024x16m8', pin 'TQ[15]', The pin 'TQ[15]' does not have a internal_power group. (LBDB-60
Warning: Line 12317, Cell 'rf_1024x16m8', pin 'RET1N', is a 'save_restore' class retention pin but is missing all recom
-982)
Warning: Line 12482, Cell 'rf_1024x16m8', pin 'STOV', The pin 'STOV' does not have a internal_power group. (LBDB-607)
Technology library 'rf_1024x16m8_nldm_tt_0p90v_0p90v_25c' read successfully
1
lc_shell> 
```

順利讀入lib檔

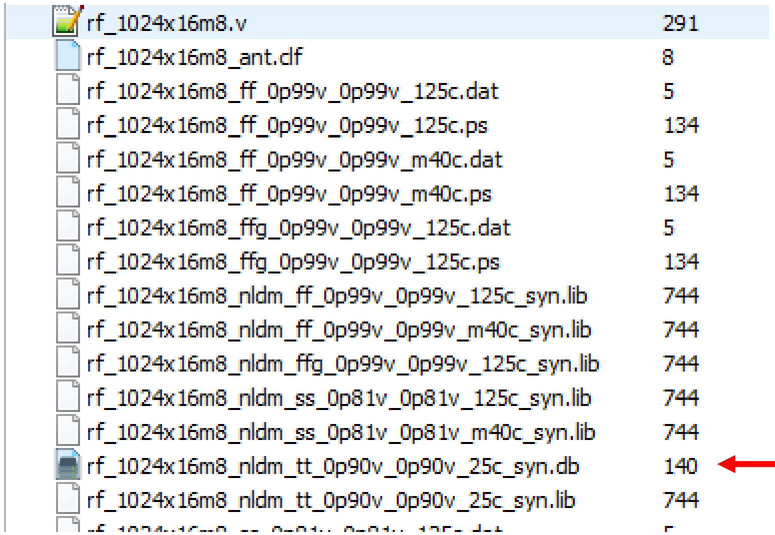


```
lc_shell> write_lib rf_1024x16m8_nldm_tt_0p90v_0p90v_25c -output rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.db
Wrote the 'rf_1024x16m8_nldm_tt_0p90v_0p90v_25c' library to '/home/m113040026/rf_1024x16m8/rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.db' success
fully
1
lc_shell> 
```

順利寫出db檔

Memory lib to db Flow（3/3）

- 轉好可在自己的資料夾看到 rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.db檔
- 要點開lib 檔案查看lib name，並非檔名。



rf_1024x16m8.v	291
rf_1024x16m8_ant.clf	8
rf_1024x16m8_ff_0p99v_0p99v_125c.dat	5
rf_1024x16m8_ff_0p99v_0p99v_125c.ps	134
rf_1024x16m8_ff_0p99v_0p99v_m40c.dat	5
rf_1024x16m8_ff_0p99v_0p99v_m40c.ps	134
rf_1024x16m8_ffg_0p99v_0p99v_125c.dat	5
rf_1024x16m8_ffg_0p99v_0p99v_125c.ps	134
rf_1024x16m8_nldm_ff_0p99v_0p99v_125c_syn.lib	744
rf_1024x16m8_nldm_ff_0p99v_0p99v_m40c_syn.lib	744
rf_1024x16m8_nldm_ffg_0p99v_0p99v_125c_syn.lib	744
rf_1024x16m8_nldm_ss_0p81v_0p81v_125c_syn.lib	744
rf_1024x16m8_nldm_ss_0p81v_0p81v_m40c_syn.lib	744
rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.db	140
rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.lib	744
rf_1024x16m8_ss_0p81v_0p81v_125c.dat	5

轉好的db檔

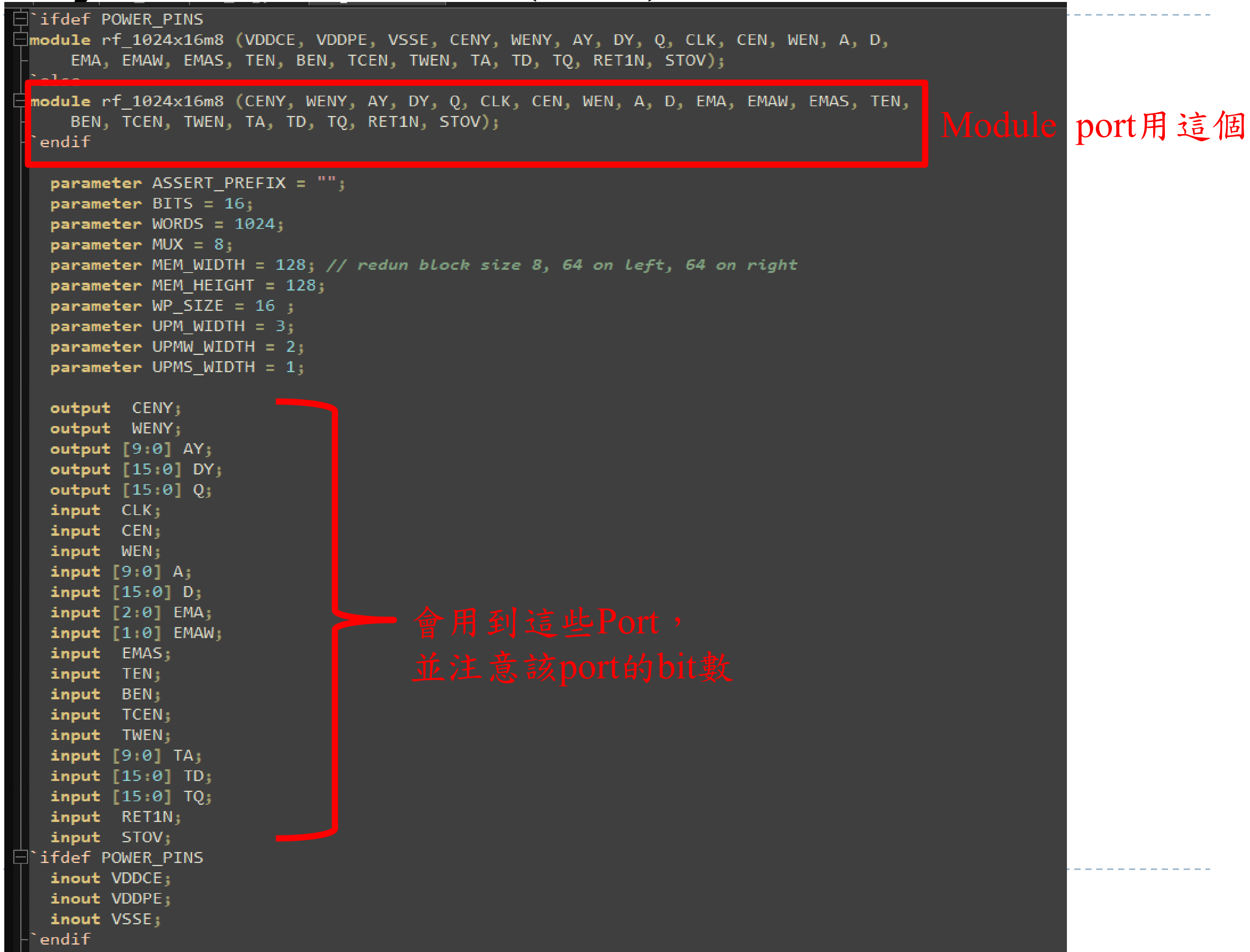
Memory Pre-sim Flow（1/8）

- Step1.設計電路
  - 設計一個verilog檔內含rf_1024x16m8的memory
  - Ex: 同時間只能做單獨讀取(1R)或單獨寫入(1W)的記憶體電路

Memory Pre-sim Flow（2/8）

- Step2.打開memory compiler產生的.v檔
- 並查看input 、output的port腳與bit數
- 注意POWER_PINS(VDDCE、VDDPE、VSSE)不用使用，該腳APR會用到。

Memory Pre-sim Flow（3/8）



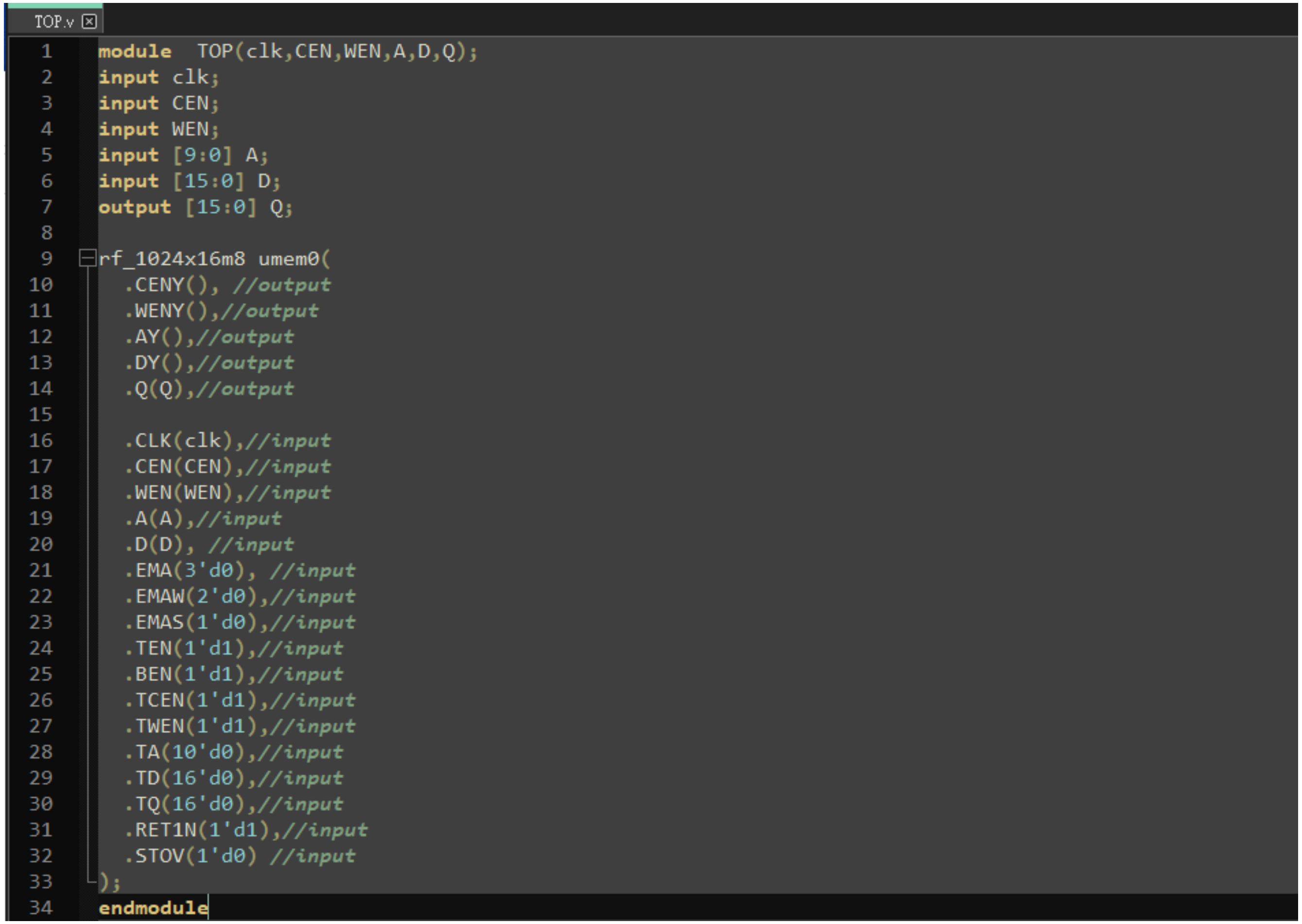
```
`ifdef POWER_PINS
module rf_1024x16m8 (VDDCE, VDDPE, VSSE, CENY, WENY, AY, DY, Q, CLK, CEN, WEN, A, D,
    EMA, EMAW, EMAS, TEN, BEN, TCEN, TWEN, TA, TD, TQ, RET1N, STOV);
`else
module rf_1024x16m8 (CENY, WENY, AY, DY, Q, CLK, CEN, WEN, A, D, EMA, EMAW, EMAS, TEN,
    BEN, TCEN, TWEN, TA, TD, TQ, RET1N, STOV);
`endif

  parameter ASSERT_PREFIX = "";
  parameter BITS = 16;
  parameter WORDS = 1024;
  parameter MUX = 8;
  parameter MEM_WIDTH = 128; // redun block size 8, 64 on left, 64 on right
  parameter MEM_HEIGHT = 128;
  parameter WP_SIZE = 16 ;
  parameter UPM_WIDTH = 3;
  parameter UPMW_WIDTH = 2;
  parameter UPMS_WIDTH = 1;

  output  CENY;
  output  WENY;
  output [9:0] AY;
  output [15:0] DY;
  output [15:0] Q;
  input  CLK;
  input  CEN;
  input  WEN;
  input [9:0] A;
  input [15:0] D;
  input [2:0] EMA;
  input [1:0] EMAW;
  input  EMAS;
  input  TEN;
  input  BEN;
  input  TCEN;
  input  TWEN;
  input [9:0] TA;
  input [15:0] TD;
  input [15:0] TQ;
  input  RET1N;
  input  STOV;
`ifdef POWER_PINS
  inout VDDCE;
  inout VDDPE;
  inout VSSE;
`endif
```

Module port用這個

會用到這些Port，
並注意該port的bit數

Memory Pre-sim Flow（4/8）

- Step 3. TOP.v 輸入以下程式碼，注意input不用不可空接，output可以



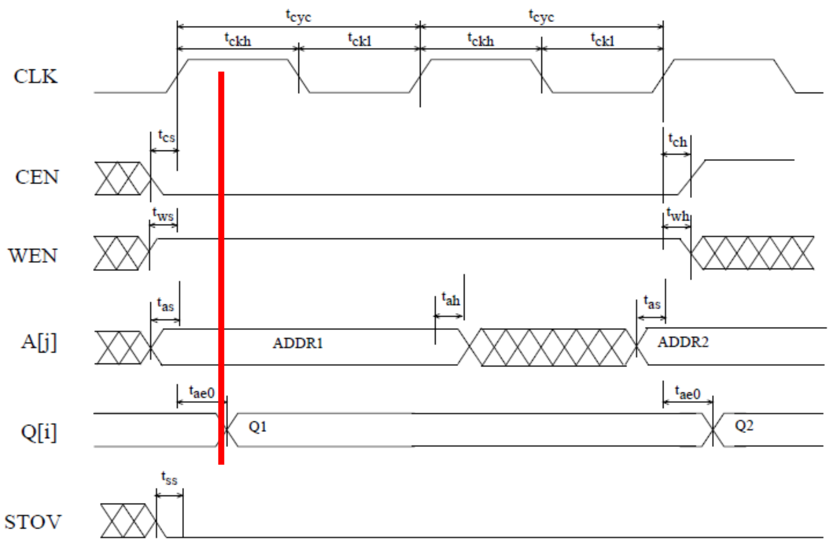
```
module  TOP(clk,CEN,WEN,A,D,Q);
input clk;
input CEN;
input WEN;
input [9:0] A;
input [15:0] D;
output [15:0] Q;

rf_1024x16m8 umem0(
  .CENY(), //output
  .WENY(),//output
  .AY(),//output
  .DY(),//output
  .Q(Q),//output

  .CLK(clk),//input
  .CEN(CEN),//input
  .WEN(WEN),//input
  .A(A),//input
  .D(D), //input
  .EMA(3'd0), //input
  .EMAW(2'd0),//input
  .EMAS(1'd0),//input
  .TEN(1'd1),//input
  .BEN(1'd1),//input
  .TCEN(1'd1),//input
  .TWEN(1'd1),//input
  .TA(10'd0),//input
  .TD(16'd0),//input
  .TQ(16'd0),//input
  .RET1N(1'd1),//input
  .STOV(1'd0) //input
);
endmodule
```

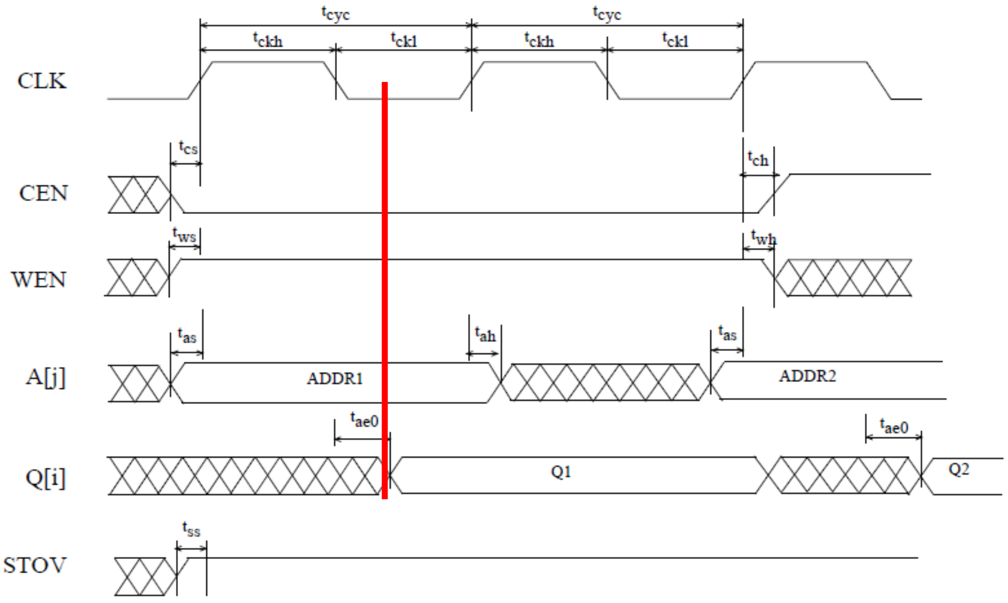
Memory Pre-sim Flow（5/8）

- Port腳說明(此範例為normal mode，可以自己調整):
- EN訊號都是active low。
- TEN、TCEN、TWEN是test專用的enable接腳 ，由於沒有要做測試電路，所以設1。
- BEN沒有要用Bypass mode，所以設1
- TA、TD、TQ 為test用的 address、data腳，沒用到設0
- EMA、EMAW、EMAS用不到設0。 沒要延遲access time。
- RET1N 用不到設1。沒做voltage retention(power down pin) 。
- STOV 設0。讓data 在posedge clk輸出 。
- 詳細說明與使用可以到/cad/CBDK/CBDK_TSMC40_Arm_f2.0/CIC/Memory/選擇的memory/rxpx/doc/裡的userguide看 。

Memory Pre-sim Flow（6/8）

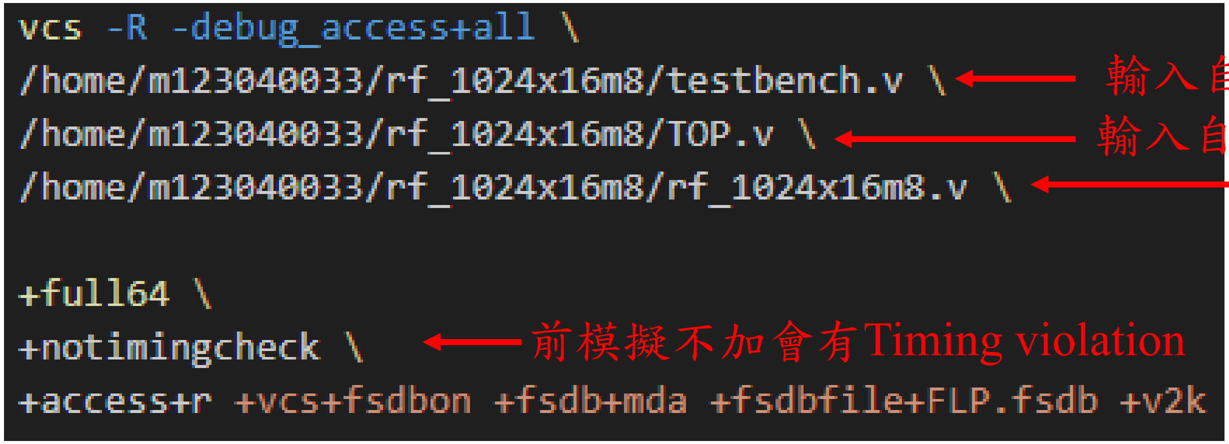
- STOV=0 與 1
- STOV =0: Q posedge clk out
- STOV =1: Q negedge clk out





Memory Pre-sim Flow（7/8）

- Step4. Pre-sim的pre_sim.sh範例，指令:source pre_sim.sh
- 記得跑pre-sim時vcs要加+notimingcheck這個指令，以免有Timing violation的問題，gate-level sim不用。



```
vcs -R -debug_access+all \
/home/m123040033/rf_1024x16m8/testbench.v \
/home/m123040033/rf_1024x16m8/TOP.v \
/home/m123040033/rf_1024x16m8/rf_1024x16m8.v \

+full64 \
+notimingcheck \
+access+r +vcs+fsdbon +fsdb+mda +fsdbfile+FLP.fsdb +v2k
```

輸入自己的tb.v與路徑

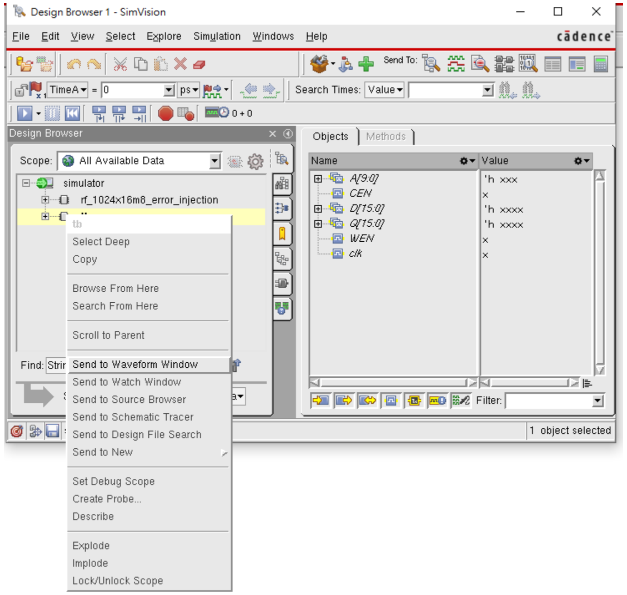
輸入自己的TOP.v與路徑

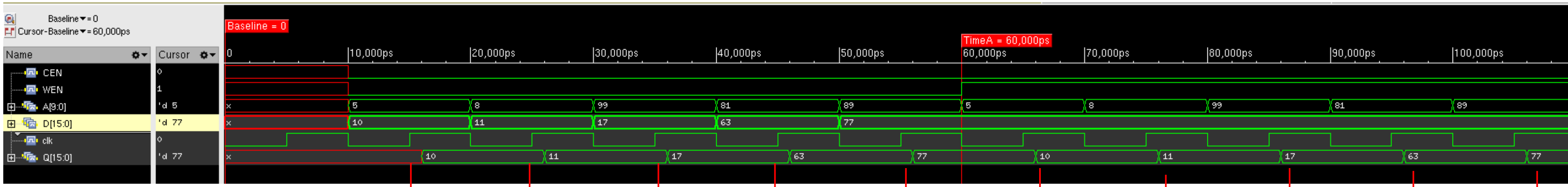
memory.v的路徑與檔案

前模擬不加會有Timing violation

Memory Pre-sim Flow（8/8）

- 波型解釋
  - 當WEN=0 → write 只看D的資訊
  - 當WEN=1 → read 只看Q的資訊
  - tb-> Send to Waveform Window -> Run





WEN=0
A=5
D=10

WEN=0
A =8
D =11

WEN=0
A =99
D =17

WEN=0
A =81
D =63

WEN=0
A =89
D =77

WEN=1
A=5
Q=10

WEN=1
A =8
Q =11

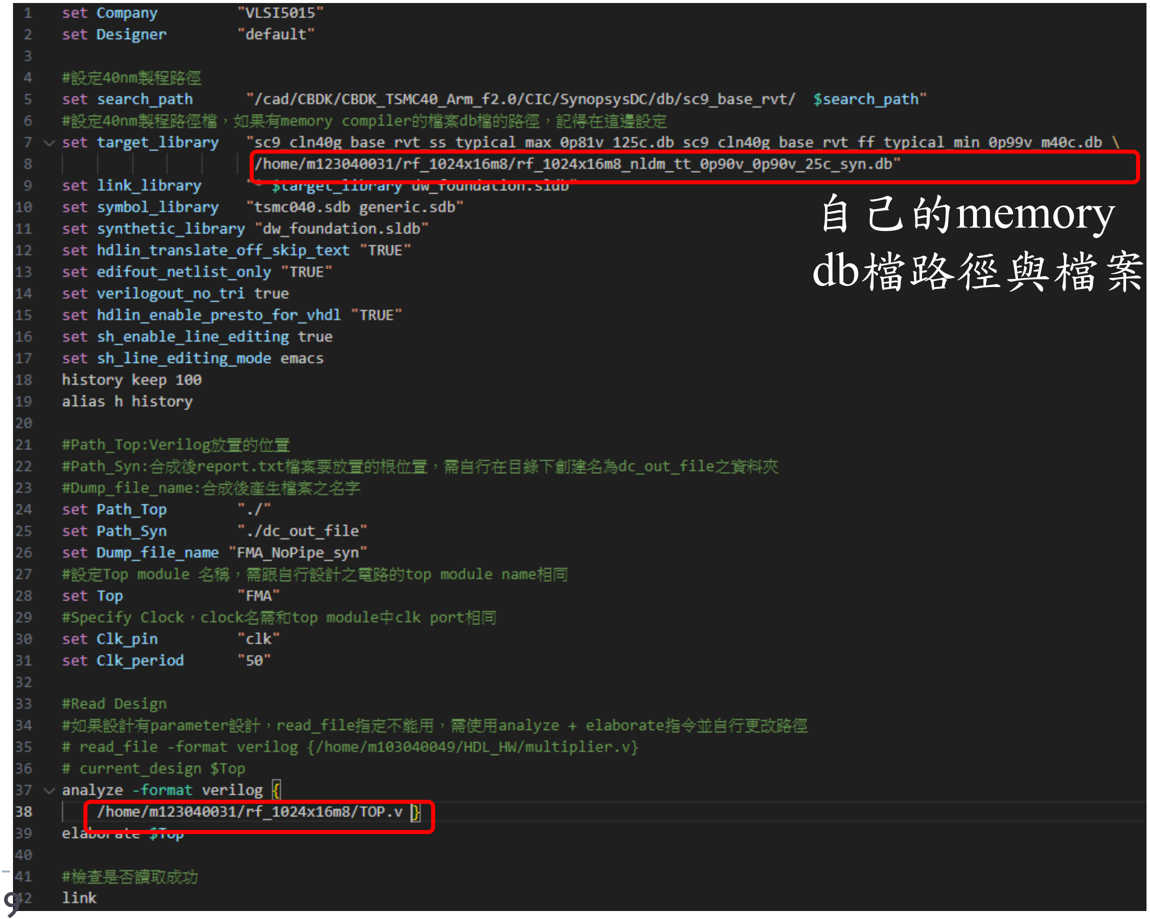
WEN=1
A =99
Q =17

WEN=1
A =81
Q =63

WEN=1
A =89
Q =77

Memory Synthesis Flow(1/6)

- Step2. 修改tcl檔 修改以下程式碼
- 1. Target library 放入剛剛產生的*.db檔路徑
- 2. 對應合成的TOP.v，進行tcl的修改，注意不要把mem.v檔拿去合

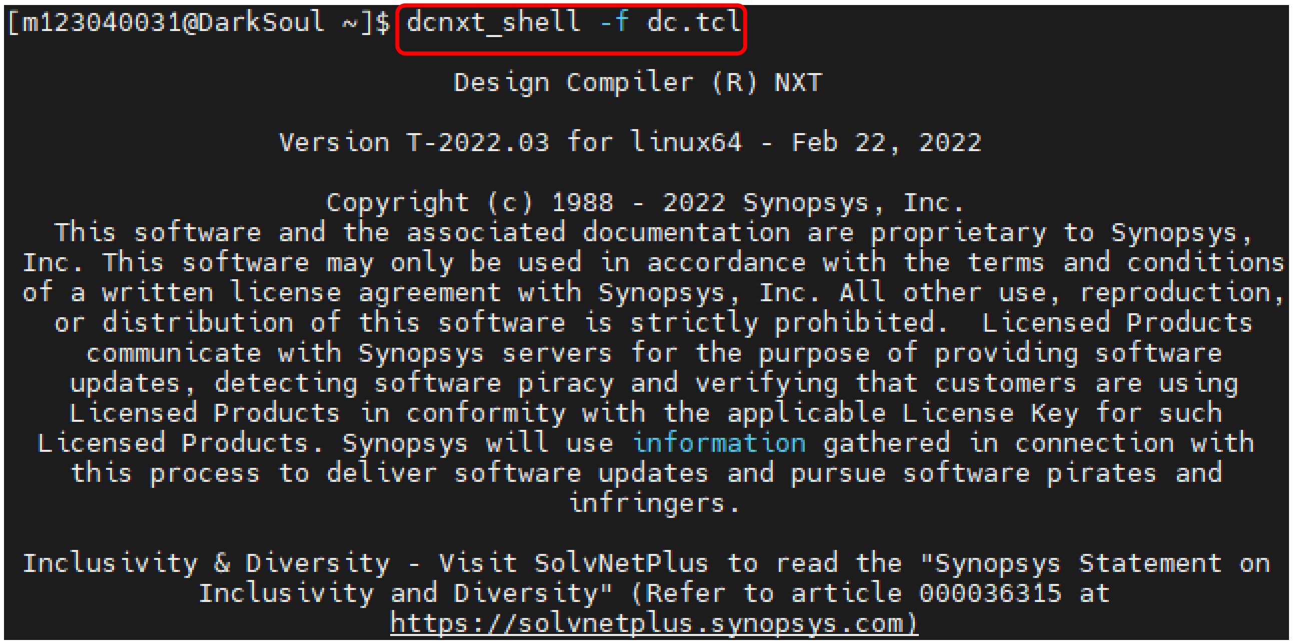


```
set Company          "VLSI5015"
set Designer         "default"

#設定40nm製程路徑
set search_path       "/cad/CBDK/CBDK_TSMC40_Arm_f2.0/CIC/SynopsysDC/db/sc9_base_rvt/  $search_path"
#設定40nm製程路徑檔，如果有memory compiler的檔案db檔的路徑，記得在這邊設定
set target_library    "sc9_cln40g_base_rvt_ss_typical_max_0p81v_125c.db sc9_cln40g_base_rvt_ff_typical_min_0p99v_m40c.db \
                      /home/m123040031/rf_1024x16m8/rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.db"
set link_library      "* $target_library dw_foundation.sldb"
set symbol_library    "tsmc040.sdb generic.sdb"
set synthetic_library "dw_foundation.sldb"
set hdlin_translate_off_skip_text "TRUE"
set edifout_netlist_only "TRUE"
set verilogout_no_tri true
set hdlin_enable_presto_for_vhdl "TRUE"
set sh_enable_line_editing true
set sh_line_editing_mode emacs
history keep 100
alias h history

#Path_Top:Verilog放置的位置
#Path_Syn:合成後report.txt檔案要放置的根位置，需自行在目錄下創建名為dc_out_file之資料夾
#Dump_file_name:合成後產生檔案之名字
set Path_Top         "./"
set Path_Syn         "./dc_out_file"
set Dump_file_name "FMA_NoPipe_syn"
#設定Top module 名稱，需跟自行設計之電路的top module name相同
set Top              "FMA"
#Specify Clock，clock名需和top module中clk port相同
set Clk_pin          "clk"
set Clk_period       "50"

#Read Design
#如果設計有parameter設計，read_file指定不能用，需使用analyze + elaborate指令並自行更改路徑
# read_file -format verilog {/home/m103040049/HDL_HW/multiplier.v}
# current_design $Top
analyze -format verilog {
    /home/m123040031/rf_1024x16m8/TOP.v }
elaborate $Top

#檢查是否讀取成功
link
```

自己的memory db檔路徑與檔案

Memory Synthesis Flow(2/6)

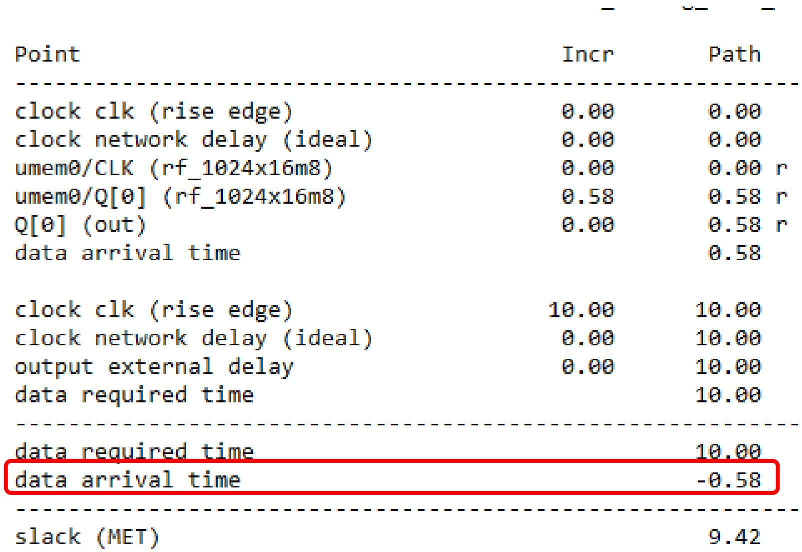
- Step3.開始合成
- 輸入指令 dcnxt_shell –f dc.tcl (對應自己取的tcl名字) 方法一



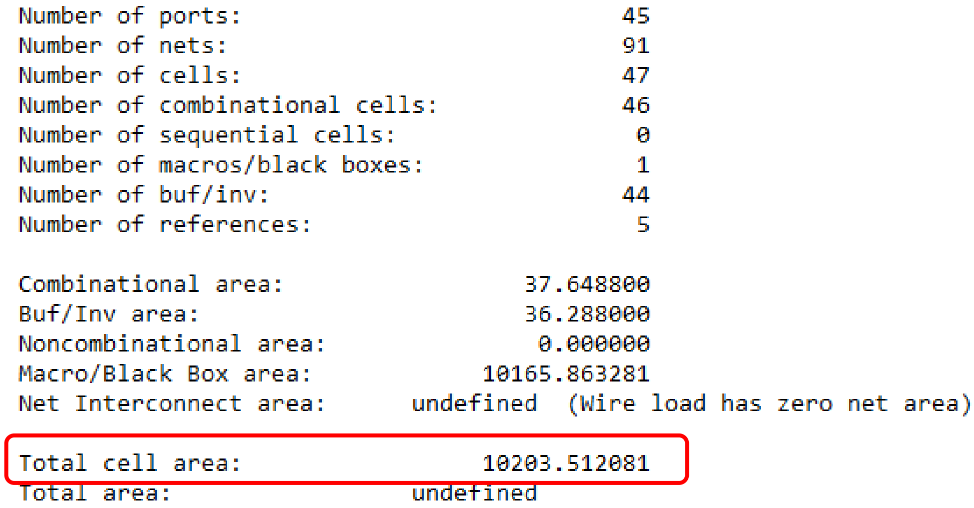
```
[m123040031@DarkSoul ~]$ dcnxt_shell -f dc.tcl

                         Design Compiler (R) NXT

              Version T-2022.03 for linux64 - Feb 22, 2022

                 Copyright (c) 1988 - 2022 Synopsys, Inc.
   This software and the associated documentation are proprietary to Synopsys,
 Inc. This software may only be used in accordance with the terms and conditions
 of a written license agreement with Synopsys, Inc. All other use, reproduction,
   or distribution of this software is strictly prohibited.  Licensed Products
     communicate with Synopsys servers for the purpose of providing software
    updates, detecting software piracy and verifying that customers are using
    Licensed Products in conformity with the applicable License Key for such
  Licensed Products. Synopsys will use information gathered in connection with
    this process to deliver software updates and pursue software pirates and
                                infringers.

 Inclusivity & Diversity - Visit SolvNetPlus to read the "Synopsys Statement on
            Inclusivity and Diversity" (Refer to article 000036315 at
                        https://solvnetplus.synopsys.com)
```

Memory Synthesis Flow(3/6)

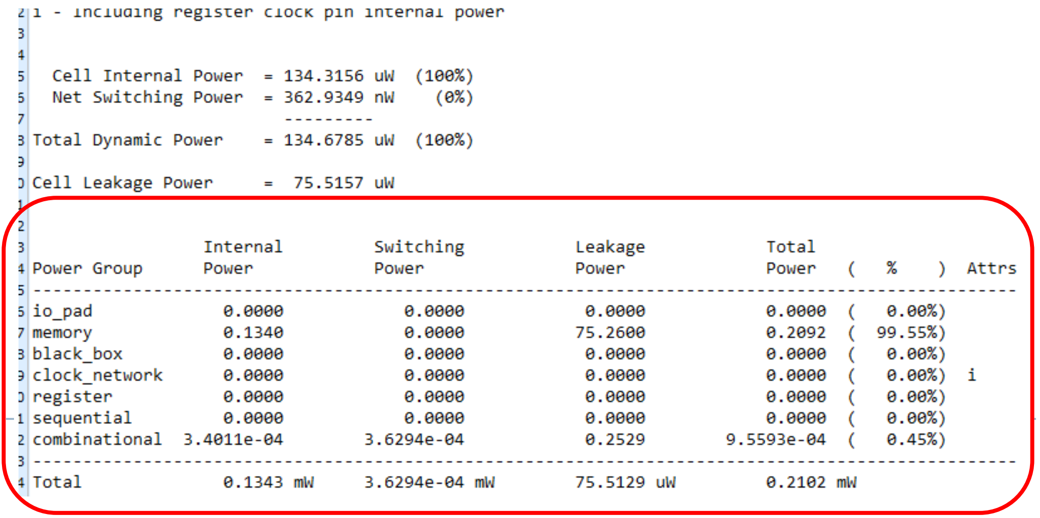
- Report Timing & Area & Power



```
Point                                       Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                       0.00       0.00
clock network delay (ideal)                 0.00       0.00
umem0/CLK (rf_1024x16m8)                    0.00       0.00 r
umem0/Q[0] (rf_1024x16m8)                   0.58       0.58 r
Q[0] (out)                                  0.00       0.58 r
data arrival time                                      0.58

clock clk (rise edge)                      10.00      10.00
clock network delay (ideal)                 0.00      10.00
output external delay                       0.00      10.00
data required time                                    10.00
-----------------------------------------------------------
data required time                                    10.00
data arrival time                                     -0.58
-----------------------------------------------------------
slack (MET)                                            9.42
```



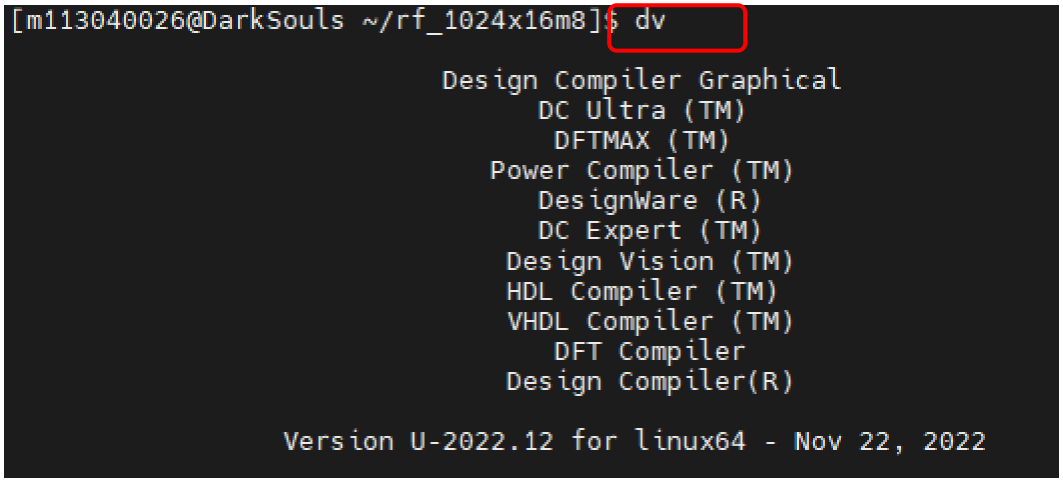
```
Number of ports:                               45
Number of nets:                                91
Number of cells:                               47
Number of combinational cells:                 46
Number of sequential cells:                     0
Number of macros/black boxes:                   1
Number of buf/inv:                             44
Number of references:                           5

Combinational area:                     37.648800
Buf/Inv area:                           36.288000
Noncombinational area:                   0.000000
Macro/Black Box area:                10165.863281
Net Interconnect area:       undefined  (Wire load has zero net area)

Total cell area:                     10203.512081
Total area:                  undefined
```



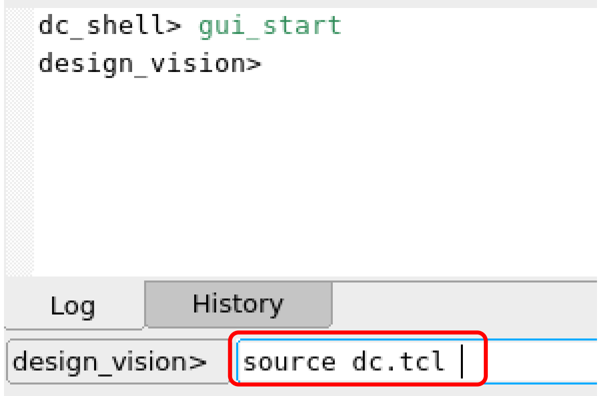
```
1 - Including register clock pin internal power


  Cell Internal Power  = 134.3156 uW  (100%)
  Net Switching Power  = 362.9349 nW    (0%)
                         ---------
Total Dynamic Power    = 134.6785 uW  (100%)

Cell Leakage Power     =  75.5157 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             0.1340           0.0000           75.2600            0.2092  (  99.55%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000            0.0000  (   0.00%)  i
register           0.0000           0.0000            0.0000            0.0000  (   0.00%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational  3.4011e-04       3.6294e-04            0.2529        9.5593e-04  (   0.45%)
--------------------------------------------------------------------------------------------------
Total              0.1343 mW    3.6294e-04 mW        75.5129 uW         0.2102 mW
```

Memory Synthesis Flow（4/6）

- Step4. (方法二)
  - 輸入指令dv → dv視窗中輸入 source dc.tcl



```
[m113040026@DarkSouls ~/rf_1024x16m8]$ dv

                           Design Compiler Graphical
                                 DC Ultra (TM)
                                  DFTMAX (TM)
                              Power Compiler (TM)
                                 DesignWare (R)
                                 DC Expert (TM)
                               Design Vision (TM)
                               HDL Compiler (TM)
                               VHDL Compiler (TM)
                                  DFT Compiler
                               Design Compiler(R)

                 Version U-2022.12 for linux64 - Nov 22, 2022
```

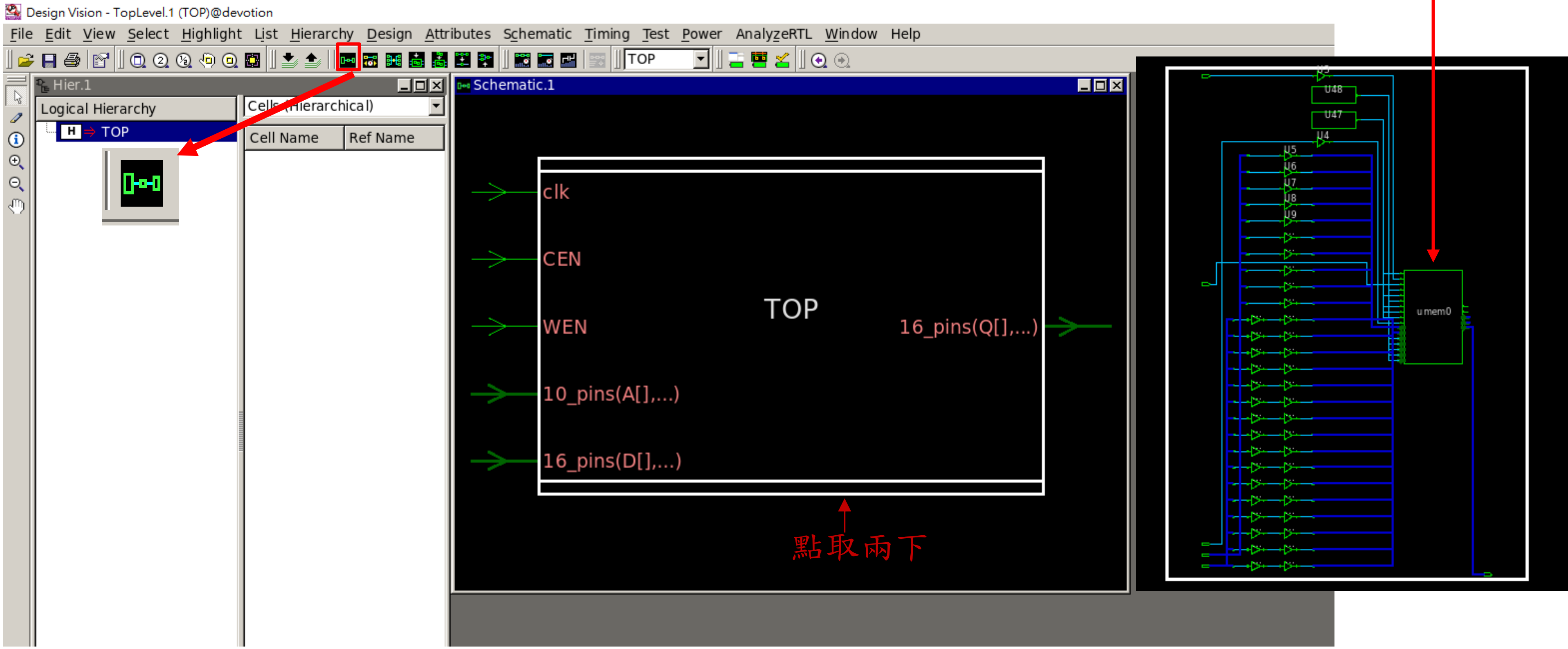


```
dc_shell> gui_start
design_vision>
```

```
design_vision> source dc.tcl
```

Memory Synthesis Flow（5/6）

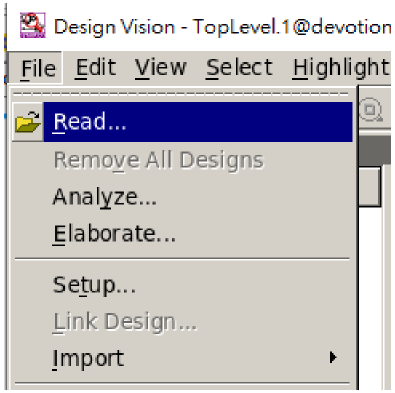
- Step4. (方法二)
  - 合成後，File → Read TOP.v
  - 觀察Schematic圖 電路是否有順利加入rf_1024x16m8



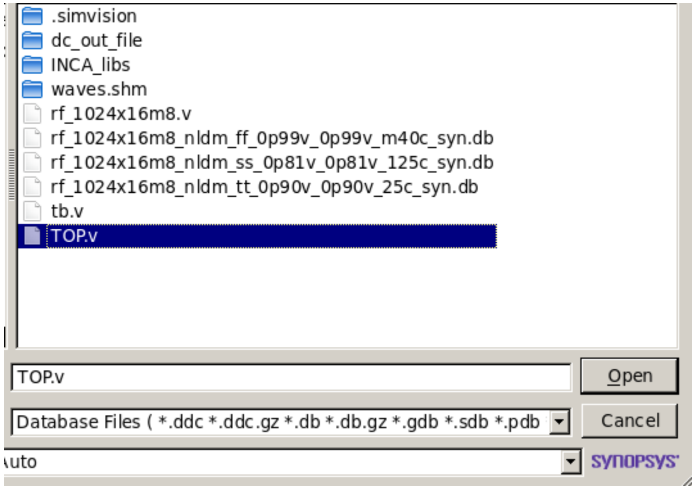
點取兩下

Memory Synthesis Flow（6/6）

Step 1.



Step 2.



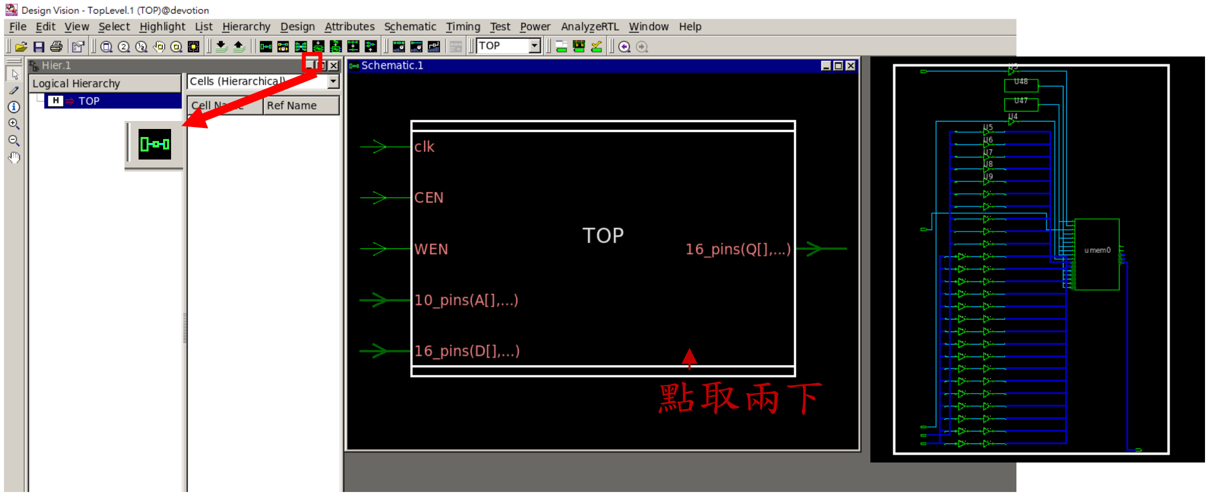
```
.simvision
dc_out_file
INCA_libs
waves.shm
rf_1024x16m8.v
rf_1024x16m8_nldm_ff_0p99v_0p99v_m40c_syn.db
rf_1024x16m8_nldm_ss_0p81v_0p81v_125c_syn.db
rf_1024x16m8_nldm_tt_0p90v_0p90v_25c_syn.db
tb.v
TOP.v
```

TOP.v

Database Files (*.ddc *.ddc.gz *.db *.db.gz *.gdb *.sdb *.pdb

uto

Step 3.



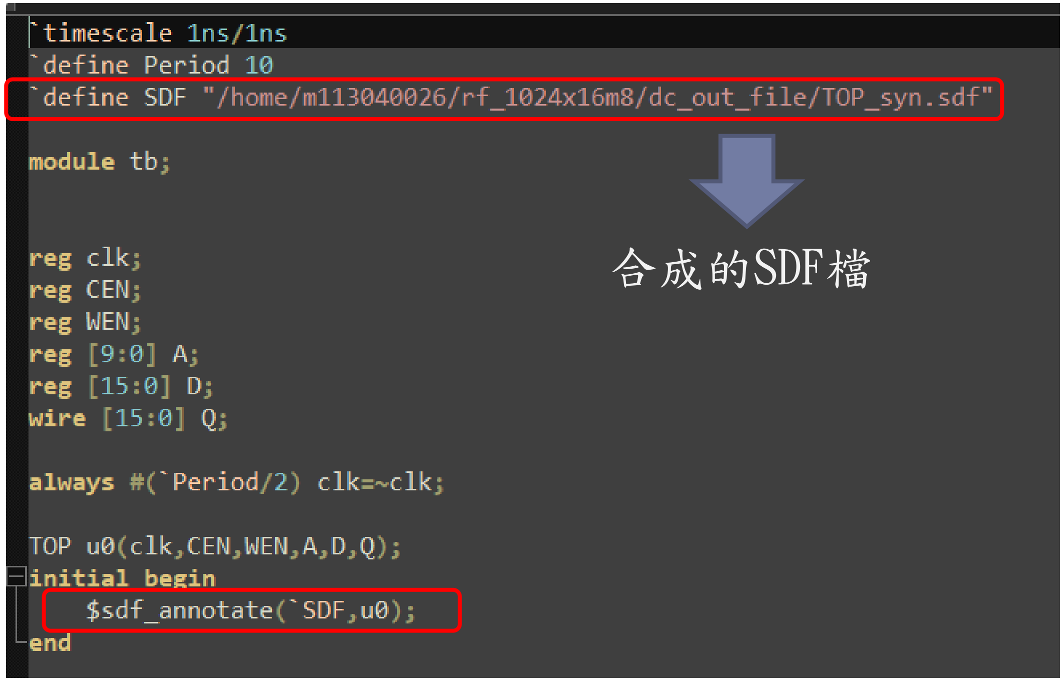
點取兩下

Memory Gate-level Simulation Flow(1/4)

- 首先需要準備檔案
  - Top_syn.v (合成後.v檔)
  - testbench.v (tb檔)
  - sc9_cln40g_base_rvt.v & sc9_cln40g_base_rvt_udp.v(cell library.v)
  - rf_1024x16m8.v (memory compiler的.v檔)
  - post_sim.sh

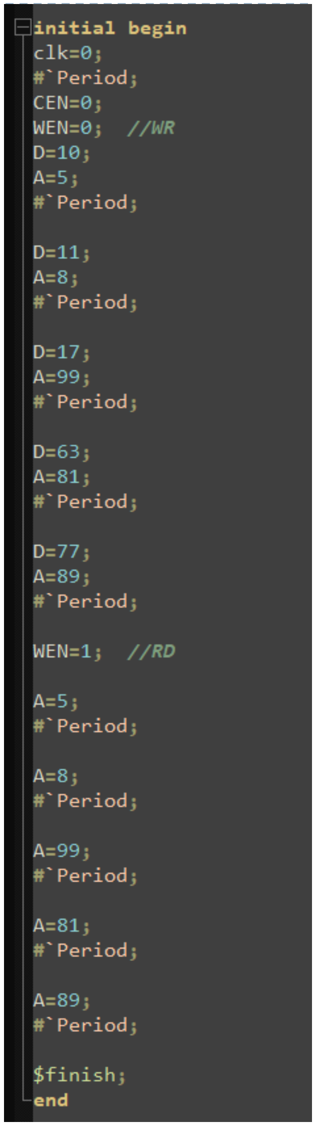
Memory Gate-level Simulation Flow(2/4)

- testbench.v 輸入以下程式碼
- 路徑改成自己的sdf路徑



```
`timescale 1ns/1ns
`define Period 10
`define SDF "/home/m113040026/rf_1024x16m8/dc_out_file/TOP_syn.sdf"

module tb;


reg clk;
reg CEN;
reg WEN;
reg [9:0] A;
reg [15:0] D;
wire [15:0] Q;

always #(`Period/2) clk=~clk;

TOP u0(clk,CEN,WEN,A,D,Q);
initial begin
    $sdf_annotate(`SDF,u0);
end
```

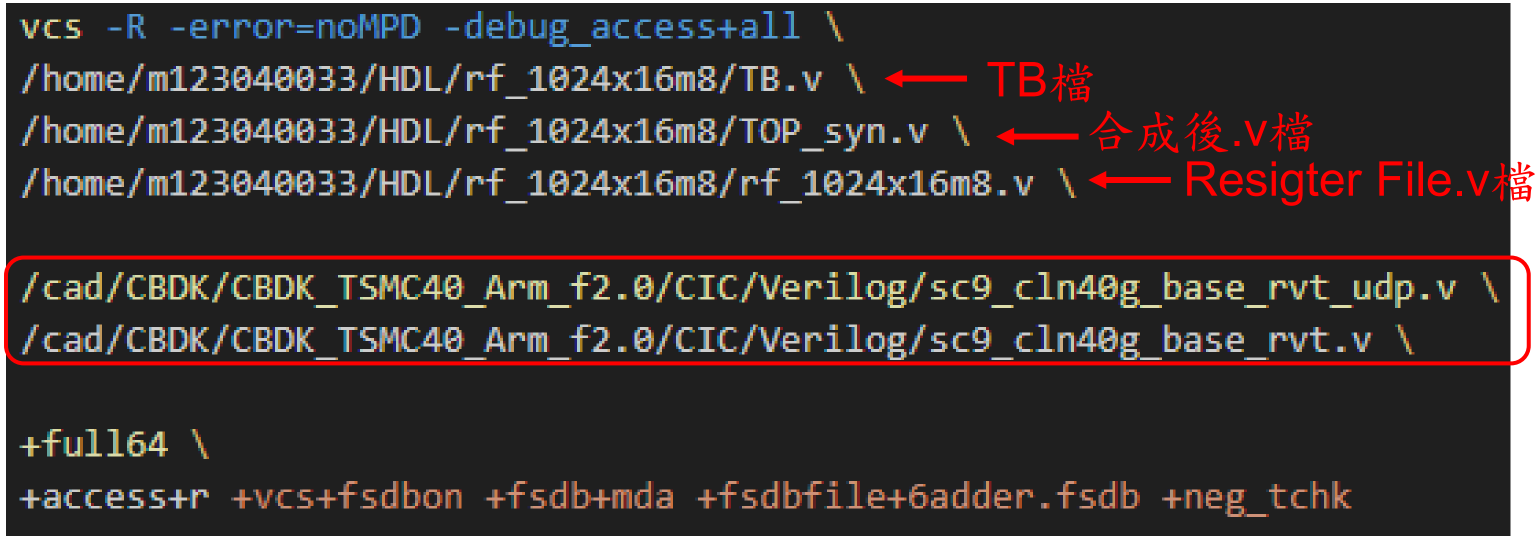
合成的SDF檔



```
initial begin
clk=0;
#`Period;
CEN=0;
WEN=0;  //WR
D=10;
A=5;
#`Period;

D=11;
A=8;
#`Period;

D=17;
A=99;
#`Period;

D=63;
A=81;
#`Period;

D=77;
A=89;
#`Period;

WEN=1;  //RD

A=5;
#`Period;

A=8;
#`Period;

A=99;
#`Period;

A=81;
#`Period;

A=89;
#`Period;

$finish;
end
```

Memory Gate-level Simulation Flow(3/4)

- 修改 post_sim.sh檔
- 指令：source post_sim.sh



```
vcs -R -error=noMPD -debug_access+all \
/home/m123040033/HDL/rf_1024x16m8/TB.v \
/home/m123040033/HDL/rf_1024x16m8/TOP_syn.v \
/home/m123040033/HDL/rf_1024x16m8/rf_1024x16m8.v \

/cad/CBDK/CBDK_TSMC40_Arm_f2.0/CIC/Verilog/sc9_cln40g_base_rvt_udp.v \
/cad/CBDK/CBDK_TSMC40_Arm_f2.0/CIC/Verilog/sc9_cln40g_base_rvt.v \

+full64 \
+access+r +vcs+fsdbon +fsdb+mda +fsdbfile+6adder.fsdb +neg_tchk
```

TB檔

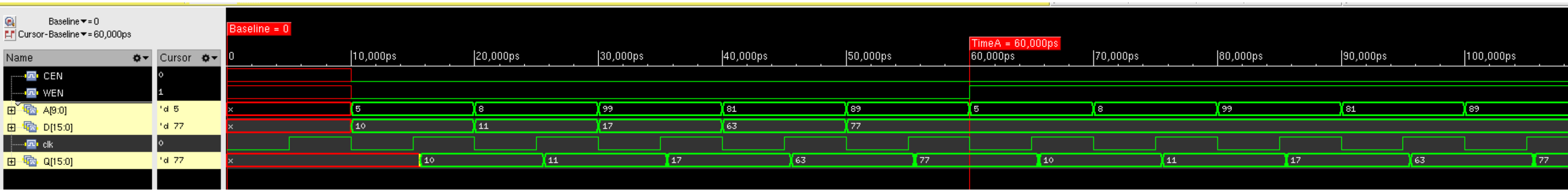
合成後.v檔

Resigter File.v檔

cell library fil

Memory Gate-level Simulation Flow（4/4）

- 波型解釋
  - 當WEN=0 → write 只看D的資訊
  - 當WEN=1 → read 只看Q的資訊
  - tb-> Send to Waveform Window -> Run



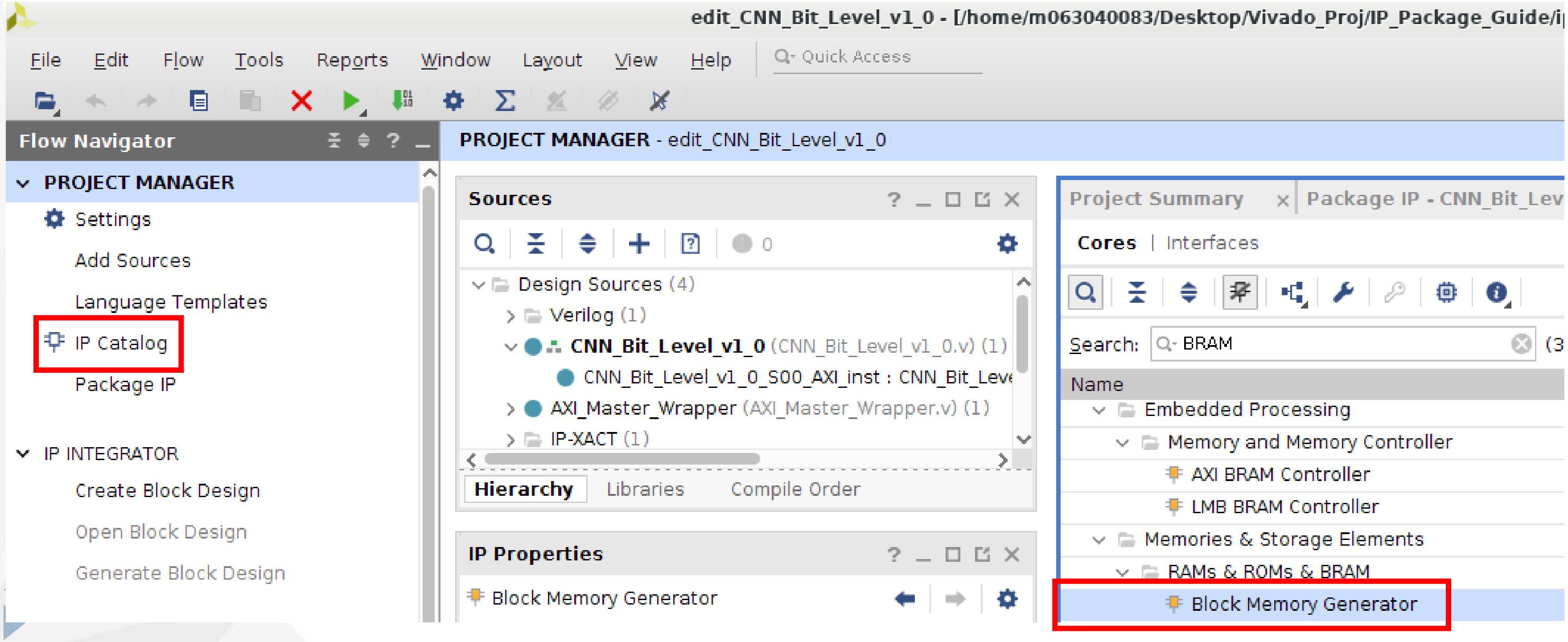
Baseline = 0
Cursor-Baseline = 60,000ps

Name	Cursor
CEN	0
WEN	1
A[9:0]	'd 5
D[15:0]	'd 77
clk	0
Q[15:0]	'd 77

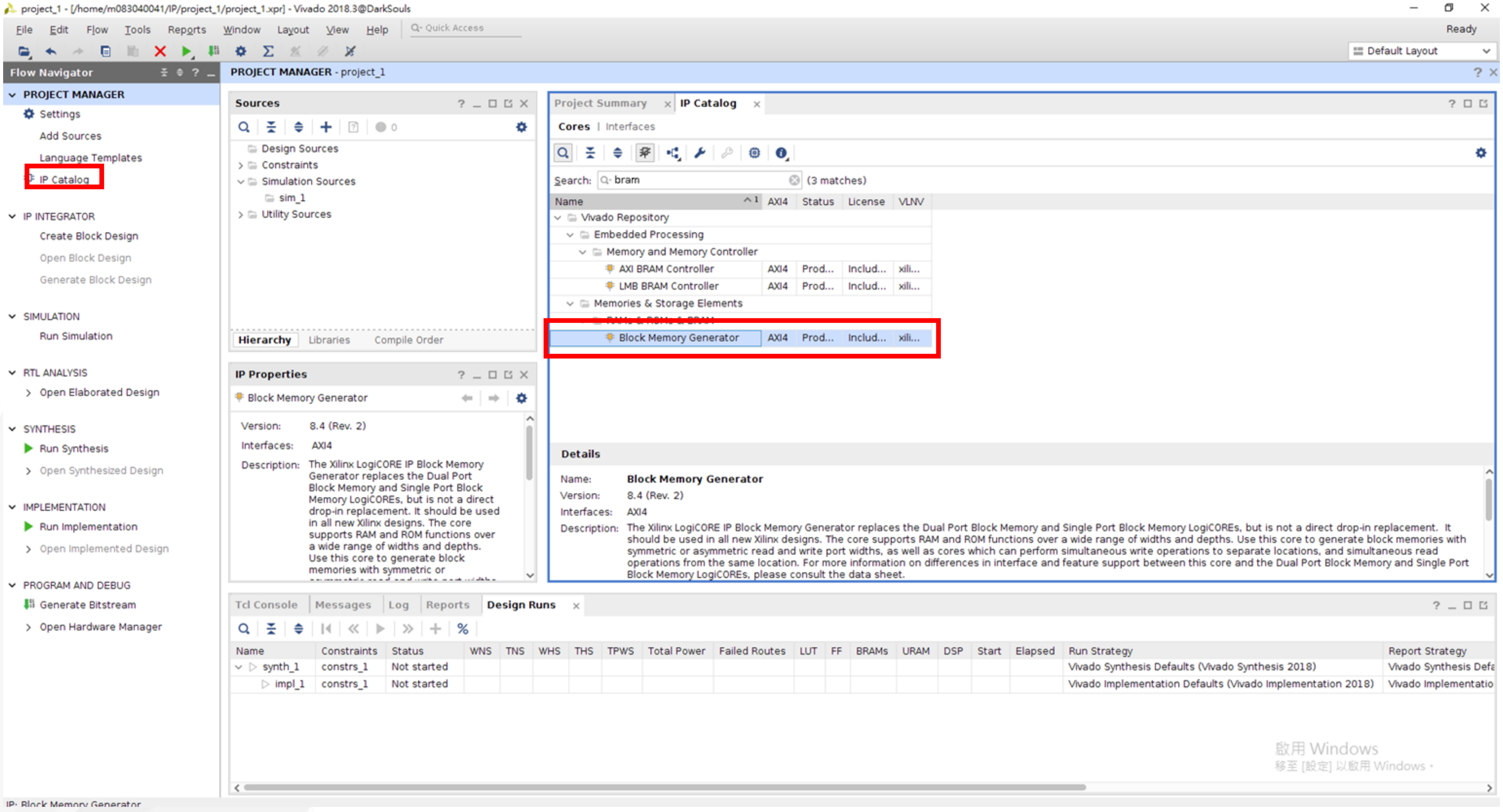
Vivado BRAM

Block memory generator(1/6)

- 如果有用到現成的IP，需於此專案重新導入
- IP Catalog -> Block memory generator (inst, in, wgt, out)



Block memory generator(2/6)



Search: bram (3 matches)

Name	AXI4	Status	License	VLNV
Vivado Repository				
Embedded Processing				
Memory and Memory Controller				
AXI BRAM Controller	AXI4	Prod...	Includ...	xili...
LMB BRAM Controller	AXI4	Prod...	Includ...	xili...
Memories & Storage Elements				
RAMs & ROMs & BRAM				
Block Memory Generator	AXI4	Prod...	Includ...	xili...

IP Properties: Block Memory Generator

Version: 8.4 (Rev. 2)

Interfaces: AXI4

Description: The Xilinx LogiCORE IP Block Memory Generator replaces the Dual Port Block Memory and Single Port Block Memory LogiCOREs, but is not a direct drop-in replacement. It should be used in all new Xilinx designs. The core supports RAM and ROM functions over a wide range of widths and depths. Use this core to generate block memories with symmetric or

Details

Name: **Block Memory Generator**

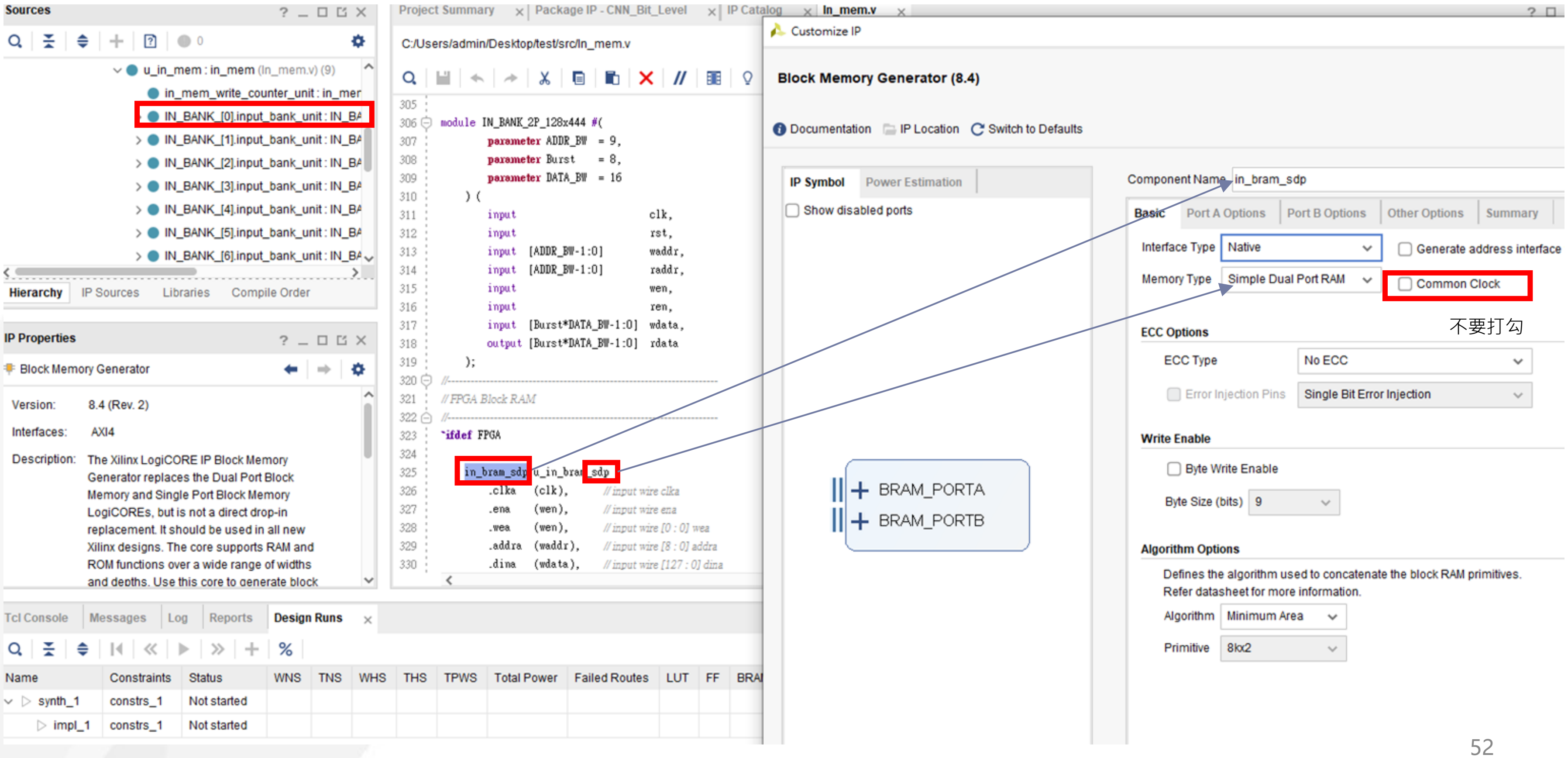
Version: 8.4 (Rev. 2)

Interfaces: AXI4

Description: The Xilinx LogiCORE IP Block Memory Generator replaces the Dual Port Block Memory and Single Port Block Memory LogiCOREs, but is not a direct drop-in replacement. It should be used in all new Xilinx designs. The core supports RAM and ROM functions over a wide range of widths and depths. Use this core to generate block memories with symmetric or asymmetric read and write port widths, as well as cores which can perform simultaneous write operations to separate locations, and simultaneous read operations from the same location. For more information on differences in interface and feature support between this core and the Dual Port Block Memory and Single Port Block Memory LogiCOREs, please consult the data sheet.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy	Report Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2018)	Vivado Synthesis Defa
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2018)	Vivado Implementatio

Block memory generator(3/6)



Sources

- u_in_mem : in_mem (In_mem.v) (9)
  - in_mem_write_counter_unit : in_mer
  - IN_BANK_[0].input_bank_unit : IN_BA
  - IN_BANK_[1].input_bank_unit : IN_BA
  - IN_BANK_[2].input_bank_unit : IN_BA
  - IN_BANK_[3].input_bank_unit : IN_BA
  - IN_BANK_[4].input_bank_unit : IN_BA
  - IN_BANK_[5].input_bank_unit : IN_BA
  - IN_BANK_[6].input_bank_unit : IN_BA

Hierarchy | IP Sources | Libraries | Compile Order

IP Properties

Block Memory Generator

Version: 8.4 (Rev. 2)

Interfaces: AXI4

Description: The Xilinx LogiCORE IP Block Memory Generator replaces the Dual Port Block Memory and Single Port Block Memory LogiCOREs, but is not a direct drop-in replacement. It should be used in all new Xilinx designs. The core supports RAM and ROM functions over a wide range of widths and depths. Use this core to generate block

Project Summary | Package IP - CNN_Bit_Level | IP Catalog | In_mem.v

C:/Users/admin/Desktop/test/src/In_mem.v

```verilog

module IN_BANK_2P_128x444 #(
        parameter ADDR_BW  = 9,
        parameter Burst    = 8,
        parameter DATA_BW  = 16
    ) (
        input                          clk,
        input                          rst,
        input  [ADDR_BW-1:0]           waddr,
        input  [ADDR_BW-1:0]           raddr,
        input                          wen,
        input                          ren,
        input  [Burst*DATA_BW-1:0]     wdata,
        output [Burst*DATA_BW-1:0]     rdata
    );
//--------------------------------------------------------
// FPGA Block RAM
//--------------------------------------------------------
`ifdef FPGA

    in_bram_sdp u_in_bram_sdp
        .clka   (clk),       // input wire clka
        .ena    (wen),       // input wire ena
        .wea    (wen),       // input wire [0 : 0] wea
        .addra  (waddr),     // input wire [8 : 0] addra
        .dina   (wdata),     // input wire [127 : 0] dina
```

Tcl Console | Messages | Log | Reports | Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAI
synth_1	constrs_1	Not started										
impl_1	constrs_1	Not started										

Customize IP

Block Memory Generator (8.4)

Documentation | IP Location | Switch to Defaults

IP Symbol | Power Estimation

□ Show disabled ports

Component Name: in_bram_sdp

Basic | Port A Options | Port B Options | Other Options | Summary

Interface Type: Native

□ Generate address interface

Memory Type: Simple Dual Port RAM

□ Common Clock

不要打勾

ECC Options

ECC Type: No ECC

□ Error Injection Pins: Single Bit Error Injection

Write Enable

□ Byte Write Enable

Byte Size (bits): 9

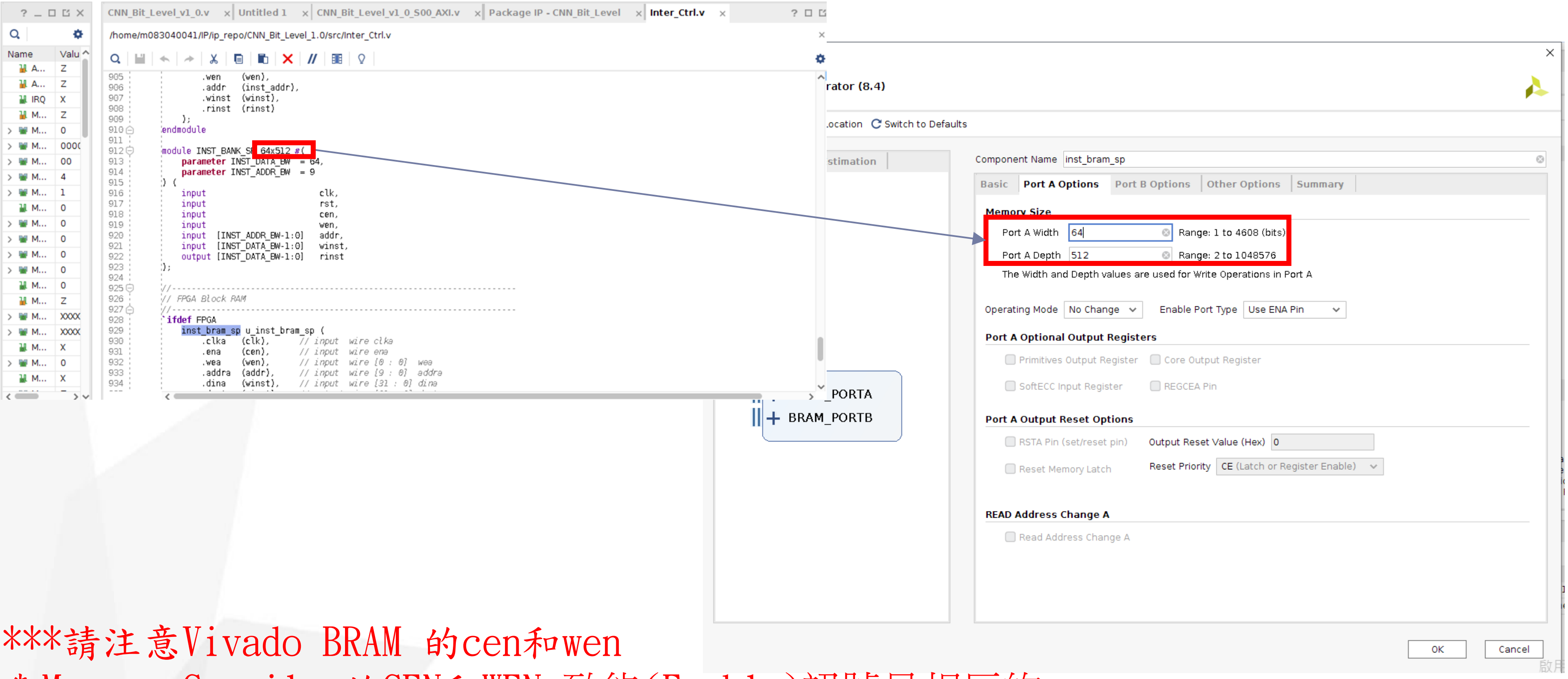
Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives. Refer datasheet for more information.

Algorithm: Minimum Area

Primitive: 8kx2

Block memory generator(4/6)



CNN_Bit_Level_v1_0.v
Untitled 1
CNN_Bit_Level_v1_0_S00_AXI.v
Package IP - CNN_Bit_Level
Inter_Ctrl.v
/home/m083040041/IP/ip_repo/CNN_Bit_Level_1.0/src/Inter_Ctrl.v

```
            .wen    (wen),
            .addr   (inst_addr),
            .winst  (winst),
            .rinst  (rinst)
        );
endmodule

module INST_BANK_SP_64x512 #(
    parameter INST_DATA_BW  = 64,
    parameter INST_ADDR_BW  = 9
) (
    input                           clk,
    input                           rst,
    input                           cen,
    input                           wen,
    input  [INST_ADDR_BW-1:0]       addr,
    input  [INST_DATA_BW-1:0]       winst,
    output [INST_DATA_BW-1:0]       rinst
);

//-----------------------------------------------------------
// FPGA Block RAM
//-----------------------------------------------------------
`ifdef FPGA
    inst_bram_sp u_inst_bram_sp (
        .clka   (clk),      // input  wire clka
        .ena    (cen),      // input  wire ena
        .wea    (wen),      // input  wire [0 : 0]  wea
        .addra  (addr),     // input  wire [9 : 0]  addra
        .dina   (winst),    // input  wire [31 : 0] dina
```

Component Name: inst_bram_sp

Memory Size

Port A Width: 64 — Range: 1 to 4608 (bits)

Port A Depth: 512 — Range: 2 to 1048576

The Width and Depth values are used for Write Operations in Port A

Operating Mode: No Change

Enable Port Type: Use ENA Pin

Port A Optional Output Registers

□ Primitives Output Register □ Core Output Register

□ SoftECC Input Register □ REGCEA Pin

Port A Output Reset Options

□ RSTA Pin (set/reset pin) Output Reset Value (Hex): 0

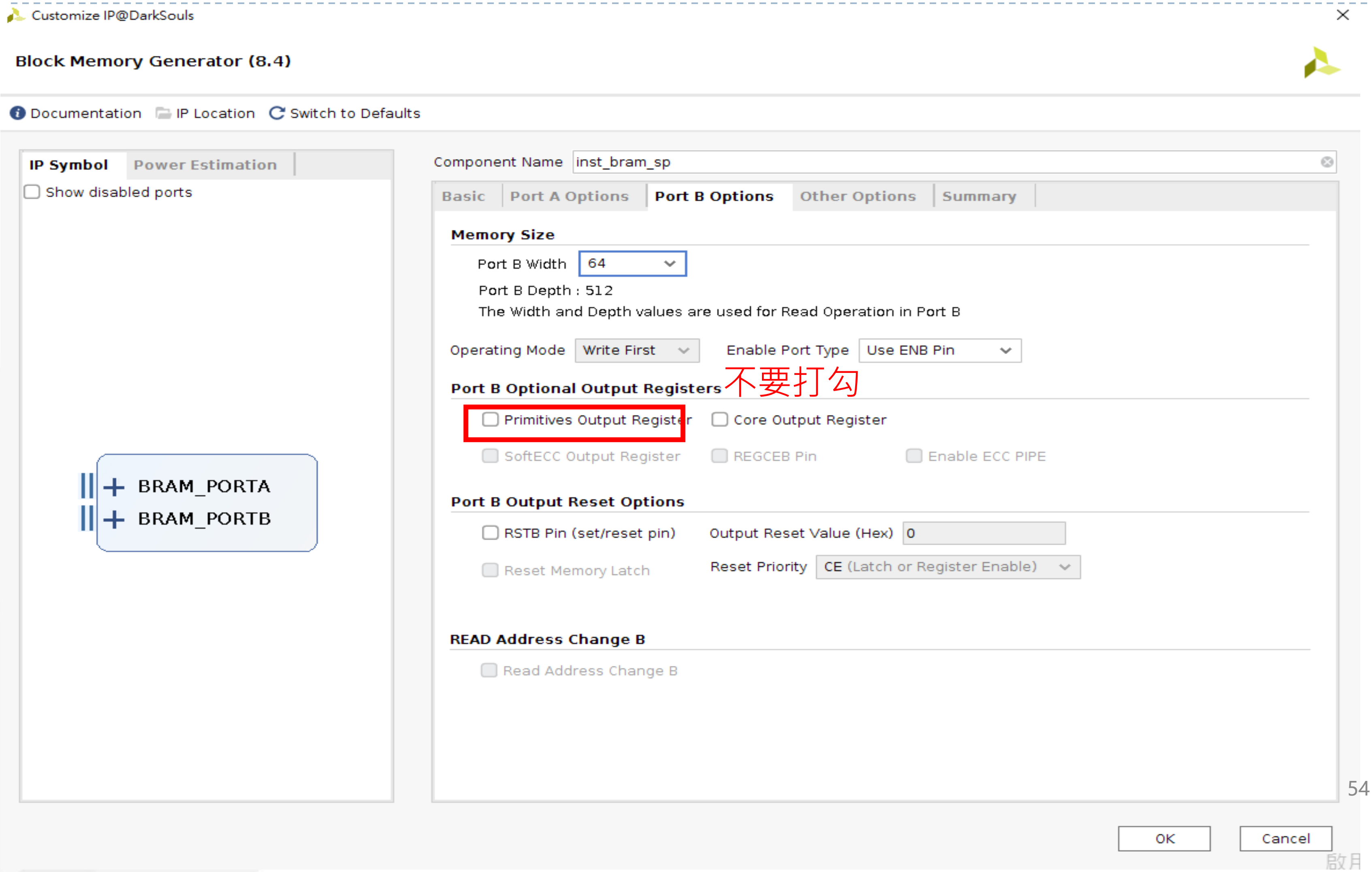
□ Reset Memory Latch Reset Priority: CE (Latch or Register Enable)

READ Address Change A

□ Read Address Change A

***請注意Vivado BRAM 的cen和wen
與Memory Compiler的CEN和WEN 致能(Enable)訊號是相反的
舉例來說:Memory Compiler 的CEN 0:Enable 1:Disable
Vivado BRAM 的CEN 0:Disable 1:Enable

Block memory generator(5/6)



Customize IP@DarkSouls

Block Memory Generator (8.4)

□ Show disabled ports

Component Name: inst_bram_sp

Memory Size

Port B Width: 64

Port B Depth : 512

The Width and Depth values are used for Read Operation in Port B

Operating Mode: Write First

Enable Port Type: Use ENB Pin

Port B Optional Output Registers 不要打勾

□ Primitives Output Register □ Core Output Register

□ SoftECC Output Register □ REGCEB Pin □ Enable ECC PIPE

Port B Output Reset Options

□ RSTB Pin (set/reset pin)

Output Reset Value (Hex): 0

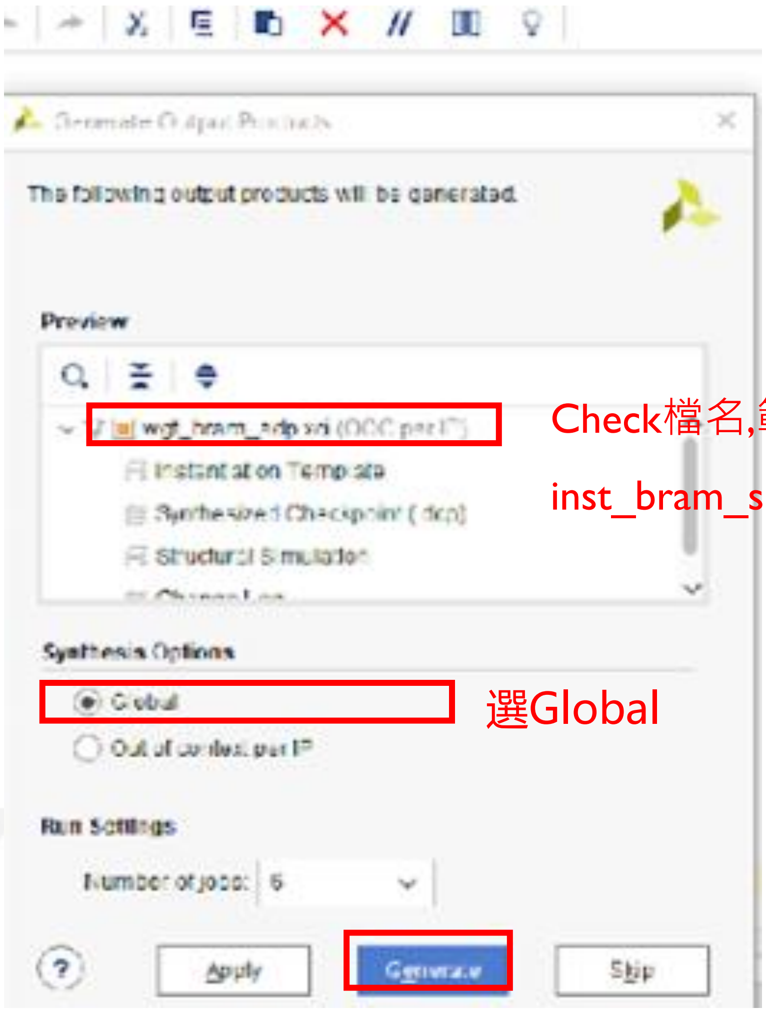
□ Reset Memory Latch

Reset Priority: CE (Latch or Register Enable)

READ Address Change B

□ Read Address Change B

Block memory generator(6/6)



Generate Output Products

The following output products will be generated.

Preview

- wgt_bram_sdp.xci (OOC per IP)
  - Instantiation Template
  - Synthesized Checkpoint (.dcp)
  - Structural Simulation
  - Change Log

Synthesis Options

● Global

○ Out of context per IP

Run Settings

Number of jobs: 6

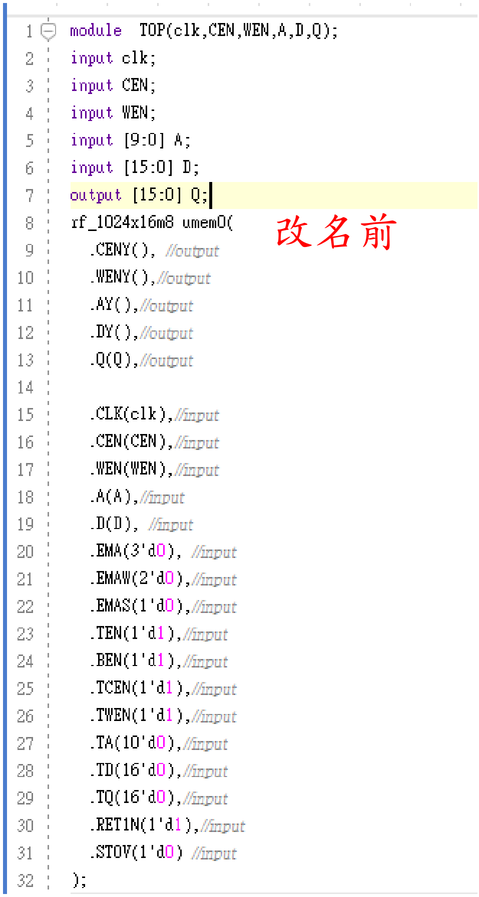
Check檔名,範例應是

inst_bram_sp.xci

選Global

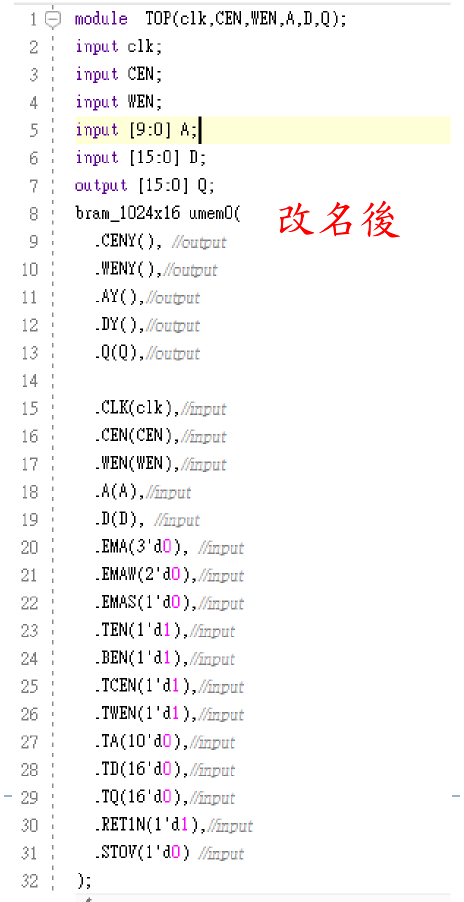
BRAM single port example (1/10)

- 使用前面flow 範例的TOP.v 與 tb.v作為範例
- Step1.匯入TOP.v 以及tb.v後由於範例取名是rf_1024x16m8，且vivado是BRAM與無mux選擇(改rf名稱可做可不做)



```
module  TOP(clk,CEN,WEN,A,D,Q);
input clk;
input CEN;
input WEN;
input [9:0] A;
input [15:0] D;
output [15:0] Q;
rf_1024x16m8 umem0(
   .CENY(), //output
   .WENY(),//output
   .AY(),//output
   .DY(),//output
   .Q(Q),//output

   .CLK(clk),//input
   .CEN(CEN),//input
   .WEN(WEN),//input
   .A(A),//input
   .D(D), //input
   .EMA(3'd0), //input
   .EMAW(2'd0),//input
   .EMAS(1'd0),//input
   .TEN(1'd1),//input
   .BEN(1'd1),//input
   .TCEN(1'd1),//input
   .TWEN(1'd1),//input
   .TA(10'd0),//input
   .TD(16'd0),//input
   .TQ(16'd0),//input
   .RET1N(1'd1),//input
   .STOV(1'd0) //input
);
```

改名前

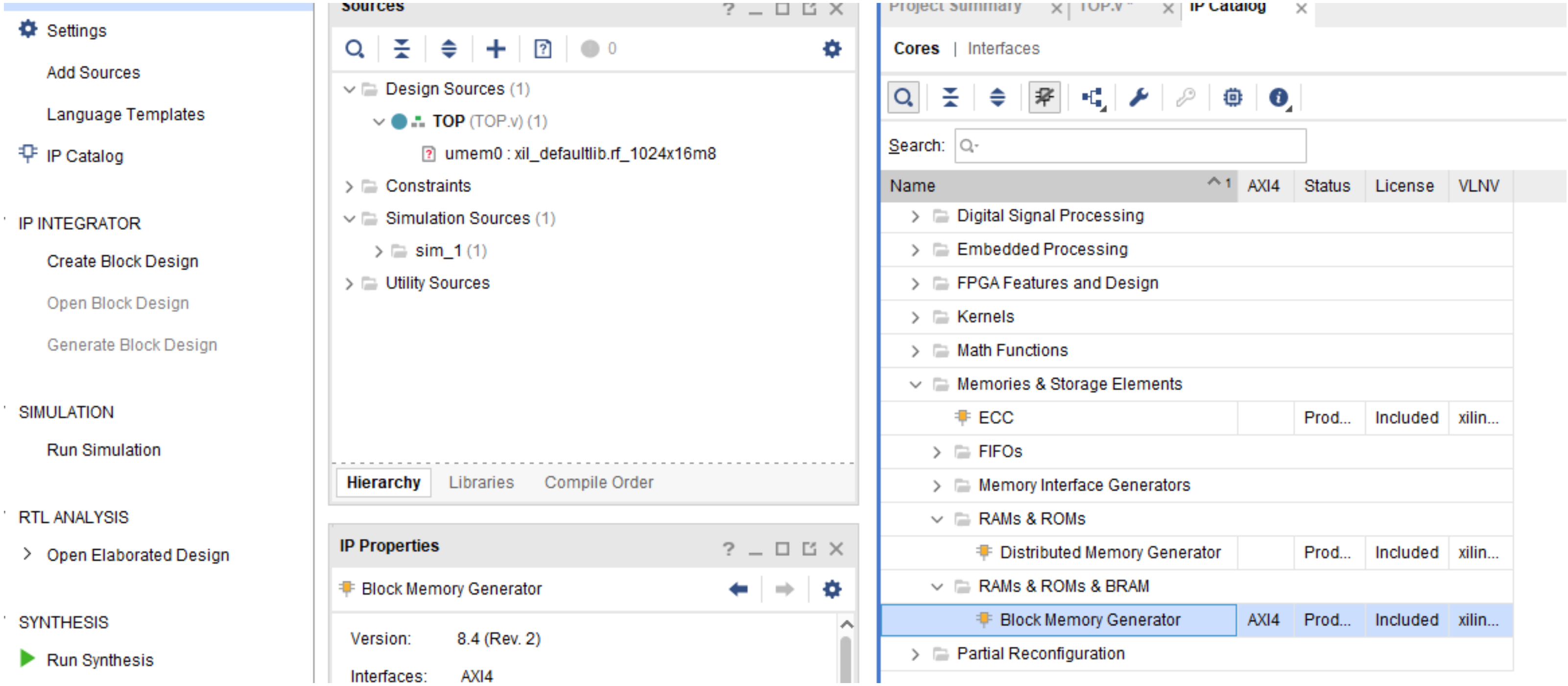


```
module  TOP(clk,CEN,WEN,A,D,Q);
input clk;
input CEN;
input WEN;
input [9:0] A;
input [15:0] D;
output [15:0] Q;
bram_1024x16 umem0(
   .CENY(), //output
   .WENY(),//output
   .AY(),//output
   .DY(),//output
   .Q(Q),//output

   .CLK(clk),//input
   .CEN(CEN),//input
   .WEN(WEN),//input
   .A(A),//input
   .D(D), //input
   .EMA(3'd0), //input
   .EMAW(2'd0),//input
   .EMAS(1'd0),//input
   .TEN(1'd1),//input
   .BEN(1'd1),//input
   .TCEN(1'd1),//input
   .TWEN(1'd1),//input
   .TA(10'd0),//input
   .TD(16'd0),//input
   .TQ(16'd0),//input
   .RET1N(1'd1),//input
   .STOV(1'd0) //input
);
```

改名後

BRAM single port example (2/10)

- Step2.點選IP Catalog找Memories &Storage Elements
- 點RAMs &ROMs &BRAM裡的Block Memory Generator



Sources:

```
Design Sources (1)
  TOP (TOP.v) (1)
    umem0 : xil_defaultlib.rf_1024x16m8
Constraints
Simulation Sources (1)
  sim_1 (1)
Utility Sources
```

IP Properties:

Block Memory Generator

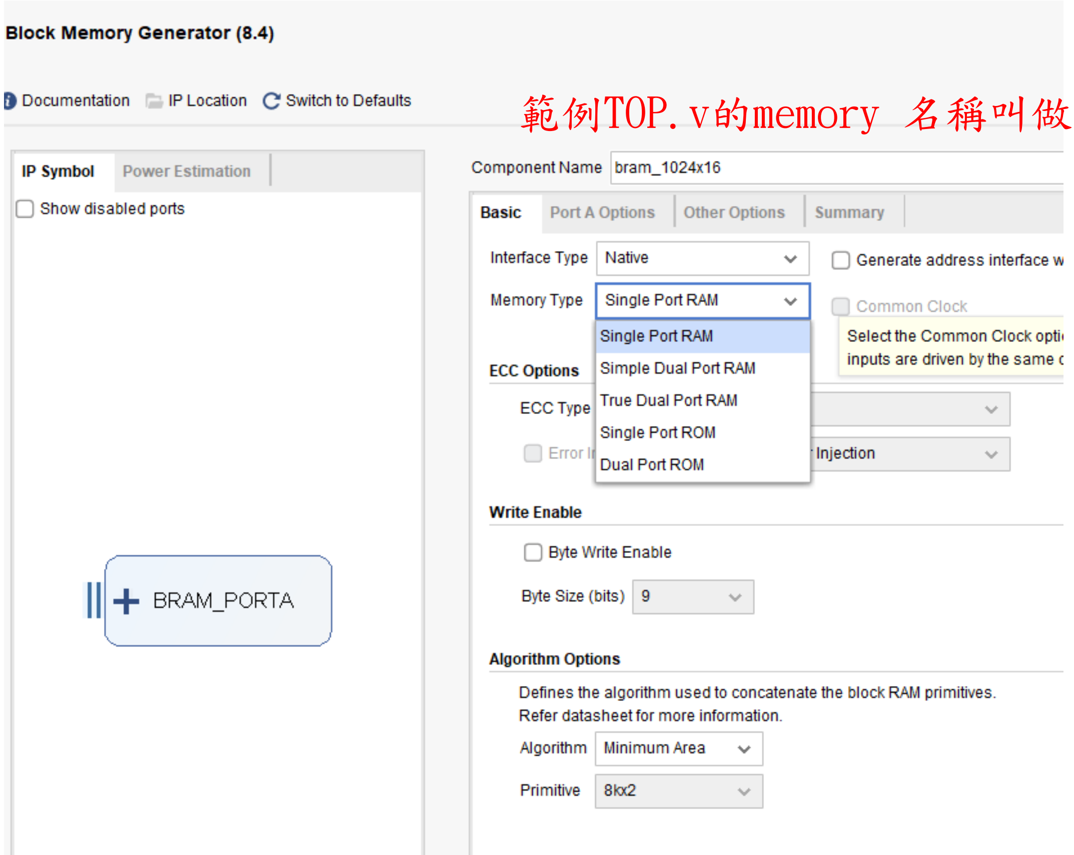
Version: 8.4 (Rev. 2)

Interfaces: AXI4

Name	AXI4	Status	License	VLNV
Digital Signal Processing				
Embedded Processing				
FPGA Features and Design				
Kernels				
Math Functions				
Memories & Storage Elements				
ECC		Prod...	Included	xilin...
FIFOs				
Memory Interface Generators				
RAMs & ROMs				
Distributed Memory Generator		Prod...	Included	xilin...
RAMs & ROMs & BRAM				
Block Memory Generator	AXI4	Prod...	Included	xilin...
Partial Reconfiguration				

BRAM single port example (3/10)

- Step3. 名稱取你TOP.v memory名稱
- Type由於此範例是single port 所以選single port



Block Memory Generator (8.4)

Documentation | IP Location | Switch to Defaults

IP Symbol | Power Estimation

☐ Show disabled ports

Component Name: bram_1024x16

Basic | Port A Options | Other Options | Summary

Interface Type: Native

☐ Generate address interface w

Memory Type: Single Port RAM

- Single Port RAM
- Simple Dual Port RAM
- True Dual Port RAM
- Single Port ROM
- Dual Port ROM

☐ Common Clock

Select the Common Clock opti
inputs are driven by the same c

ECC Options

ECC Type

☐ Error I

Injection

Write Enable

☐ Byte Write Enable

Byte Size (bits): 9

Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.

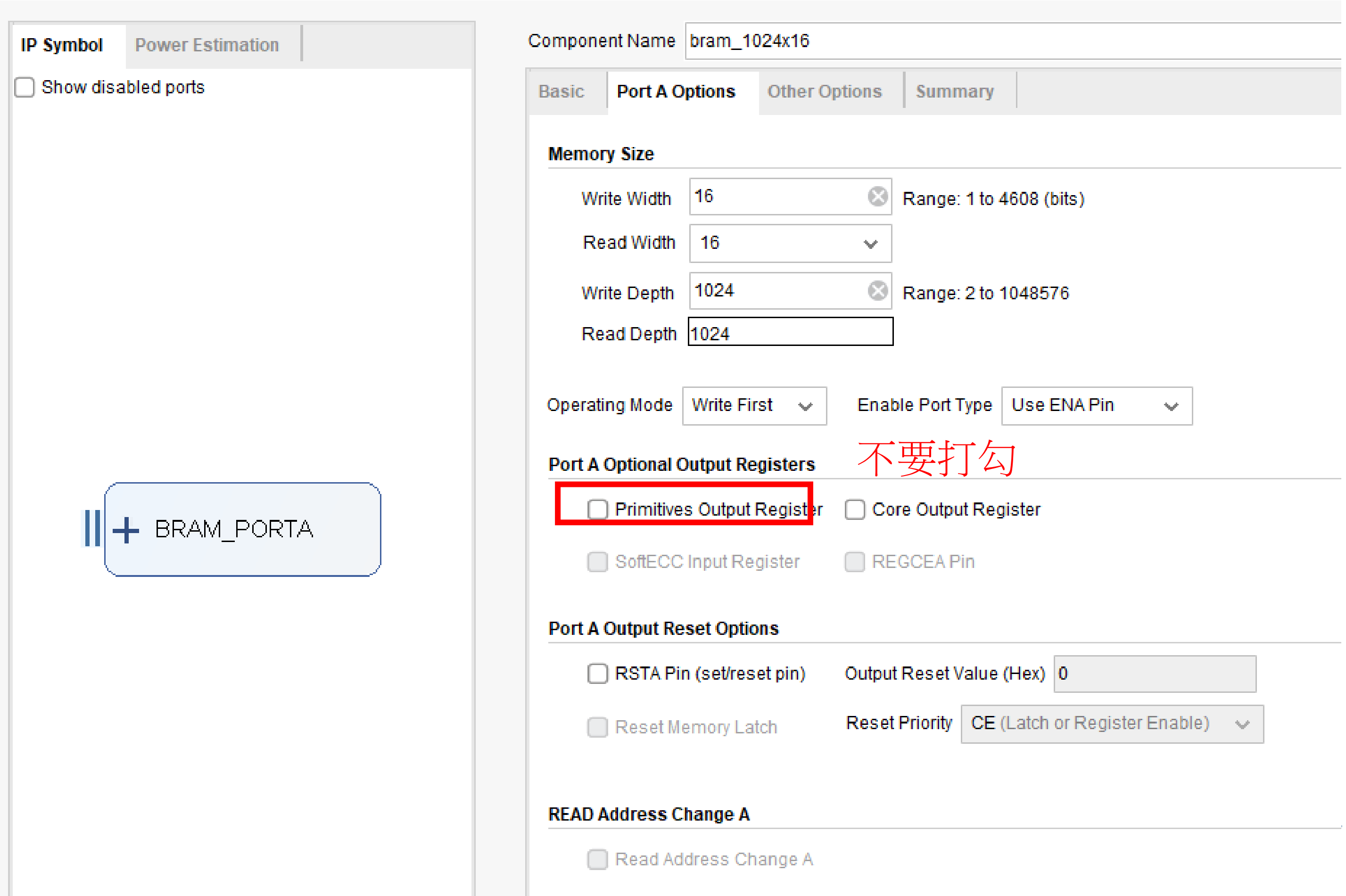
Algorithm: Minimum Area

Primitive: 8kx2

範例TOP.v的memory 名稱叫做bram_1024x16

BRAM single port example (4/10)

- Step4.Port A Options的Width 與depth為16與1024
- 若是Dual port 則有Port B要設定，都好了按OK



IP Symbol | Power Estimation

□ Show disabled ports

Component Name bram_1024x16

Basic | Port A Options | Other Options | Summary

Memory Size

Write Width 16 Range: 1 to 4608 (bits)

Read Width 16

Write Depth 1024 Range: 2 to 1048576

Read Depth 1024

Operating Mode Write First

Enable Port Type Use ENA Pin

Port A Optional Output Registers

不要打勾

□ Primitives Output Register □ Core Output Register

□ SoftECC Input Register □ REGCEA Pin

Port A Output Reset Options

□ RSTA Pin (set/reset pin) Output Reset Value (Hex) 0

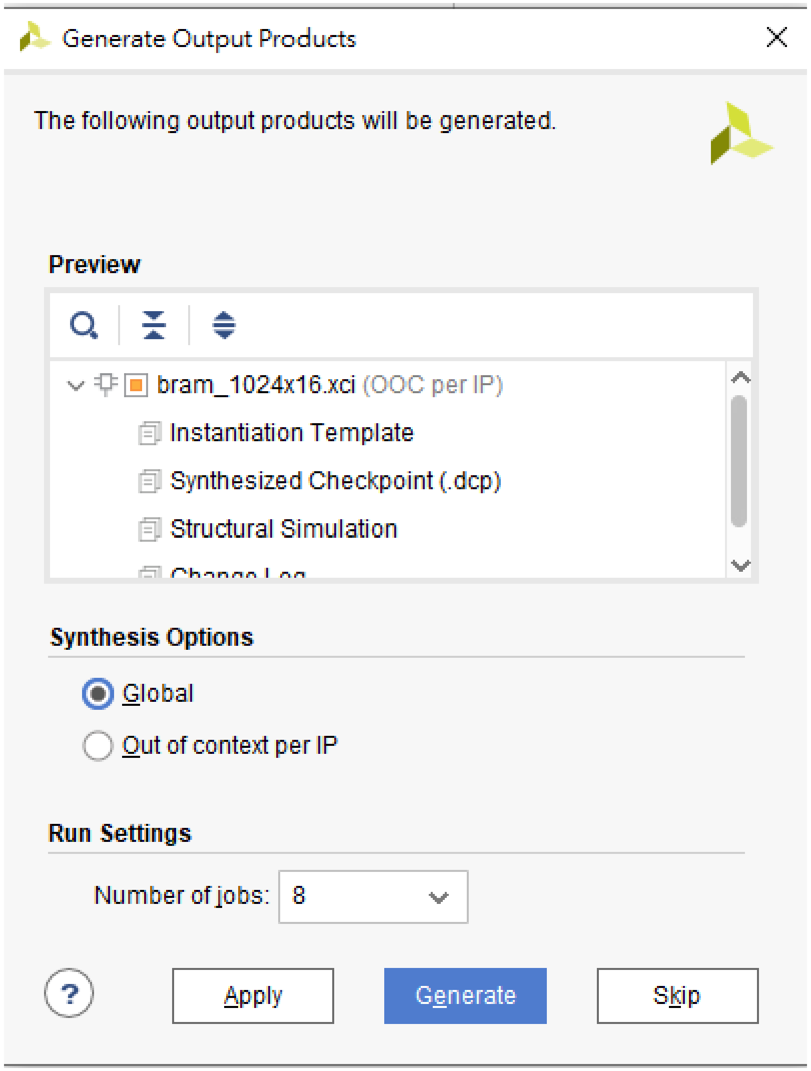
□ Reset Memory Latch Reset Priority CE (Latch or Register Enable)

READ Address Change A

□ Read Address Change A

BRAM single port example (5/10)

- Step5.選Global 按Generate



Generate Output Products

The following output products will be generated.

Preview

- bram_1024x16.xci (OOC per IP)
  - Instantiation Template
  - Synthesized Checkpoint (.dcp)
  - Structural Simulation
  - Change Log

Synthesis Options

● Global

○ Out of context per IP

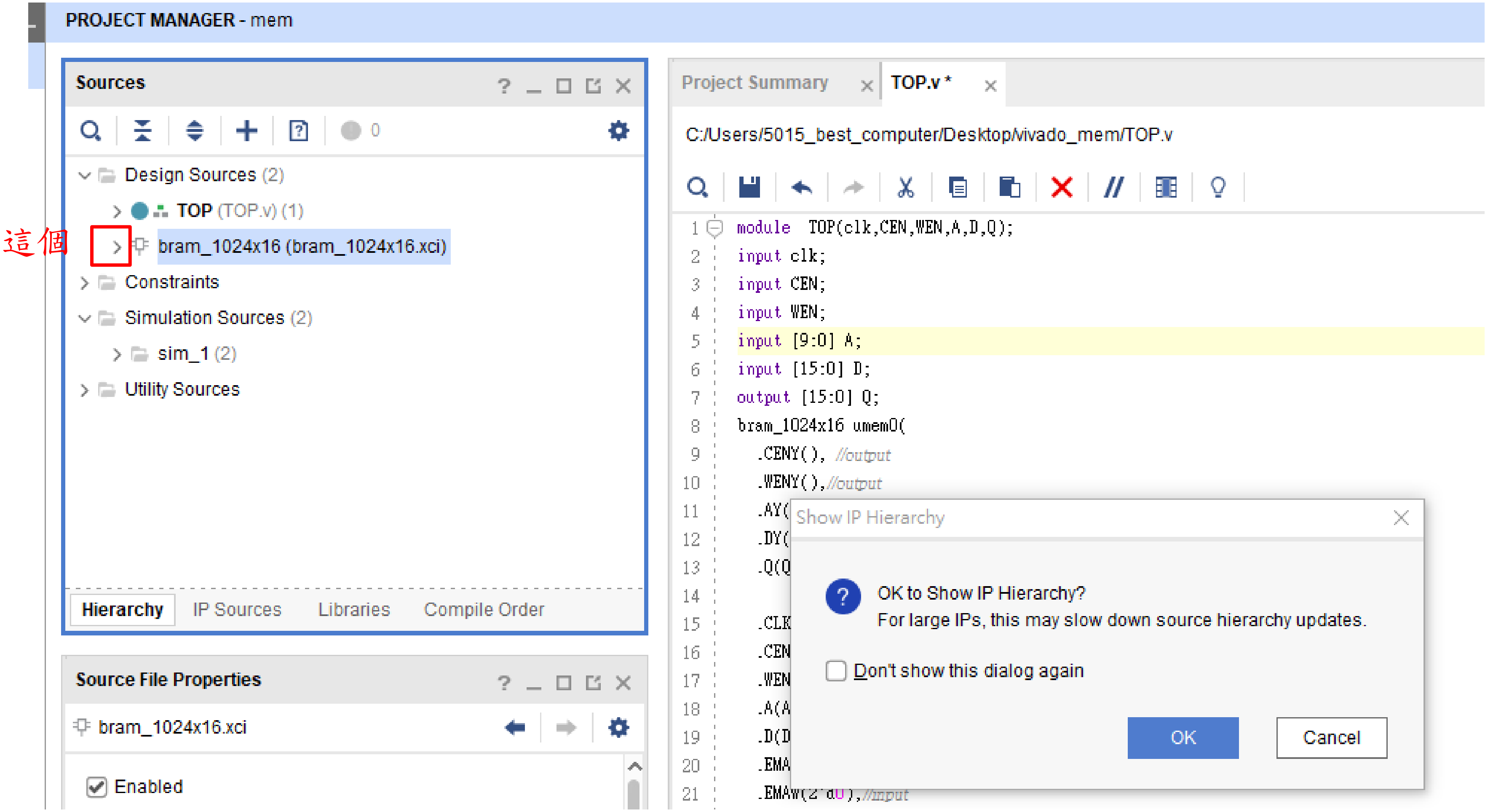
Run Settings

Number of jobs: 8

BRAM single port example (6/10)

- Step6.點bram旁邊的>，跑出Show IP Hierarchy並按OK。

點這個



C:/Users/5015_best_computer/Desktop/vivado_mem/TOP.v

```
module  TOP(clk,CEN,WEN,A,D,Q);
input clk;
input CEN;
input WEN;
input [9:0] A;
input [15:0] D;
output [15:0] Q;
bram_1024x16 umem0(
   .CENY(), //output
   .WENY(),//output
   .AY(
   .DY(
   .Q(Q

   .CLK
   .CEN
   .WEN
   .A(A
   .D(D
   .EMA
   .EMAW(2'd0),//input
```

Show IP Hierarchy

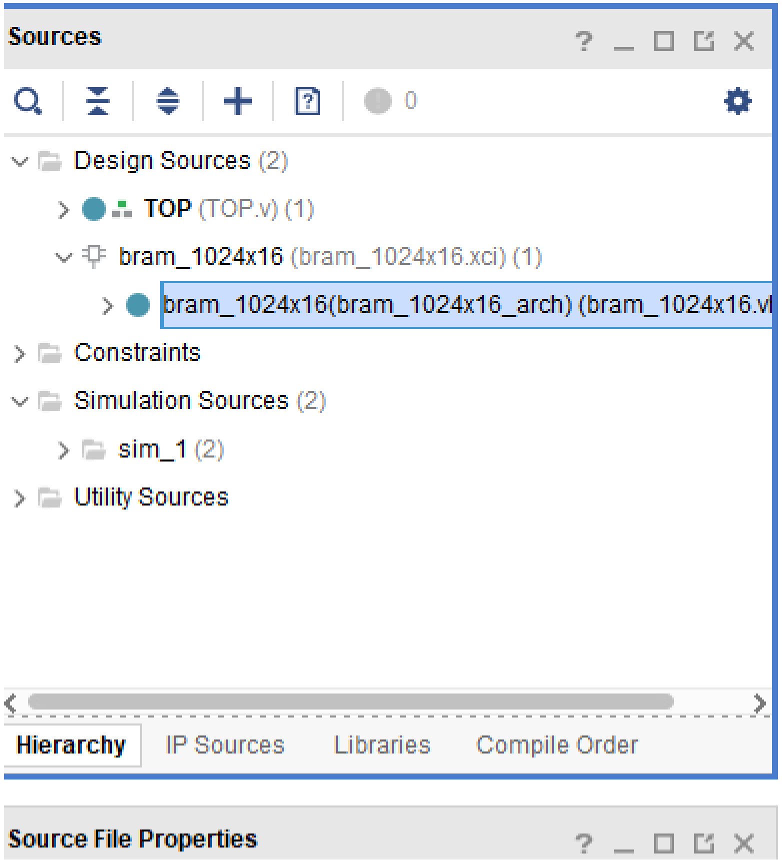
OK to Show IP Hierarchy?
For large IPs, this may slow down source hierarchy updates.

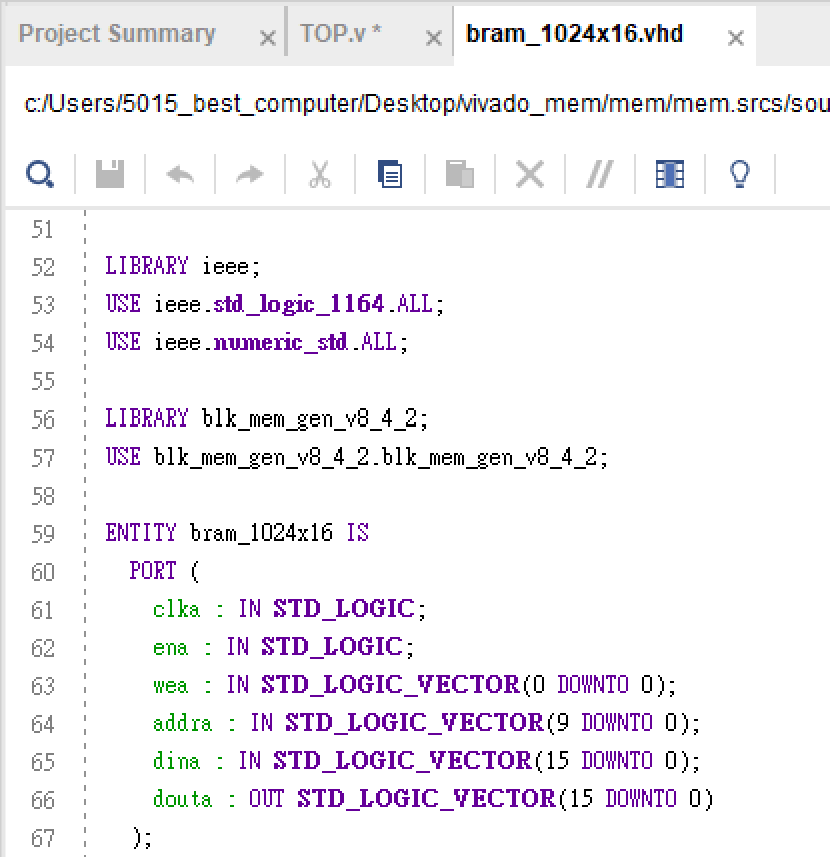
□ Don't show this dialog again

OK   Cancel

BRAM single port example (7/10)

- Step7.打開bram1024x16.vhd查看接腳名稱並把TOP.v的接腳更改



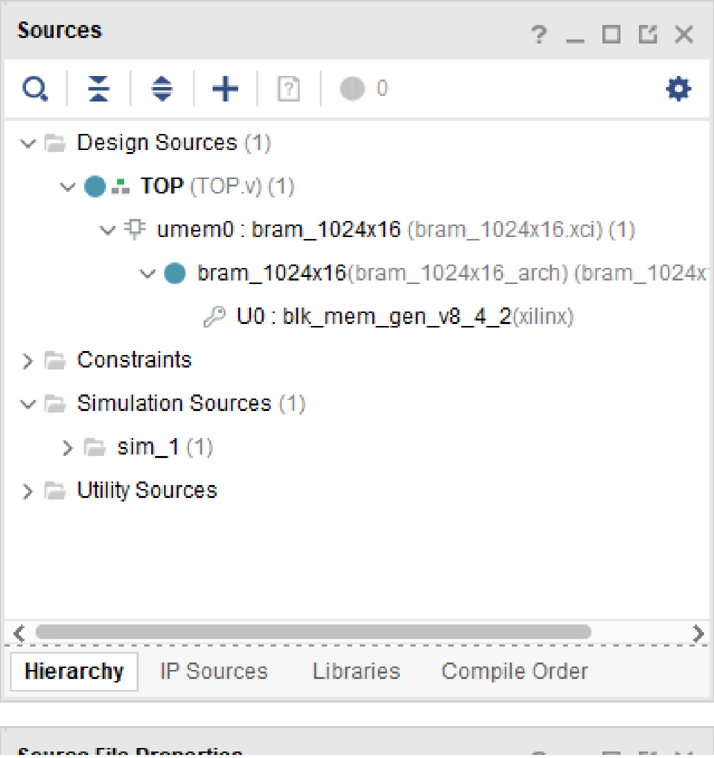


c:/Users/5015_best_computer/Desktop/vivado_mem/mem/mem.srcs/sou

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

LIBRARY blk_mem_gen_v8_4_2;
USE blk_mem_gen_v8_4_2.blk_mem_gen_v8_4_2;

ENTITY bram_1024x16 IS
  PORT (
    clka : IN STD_LOGIC;
    ena : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
  );
```

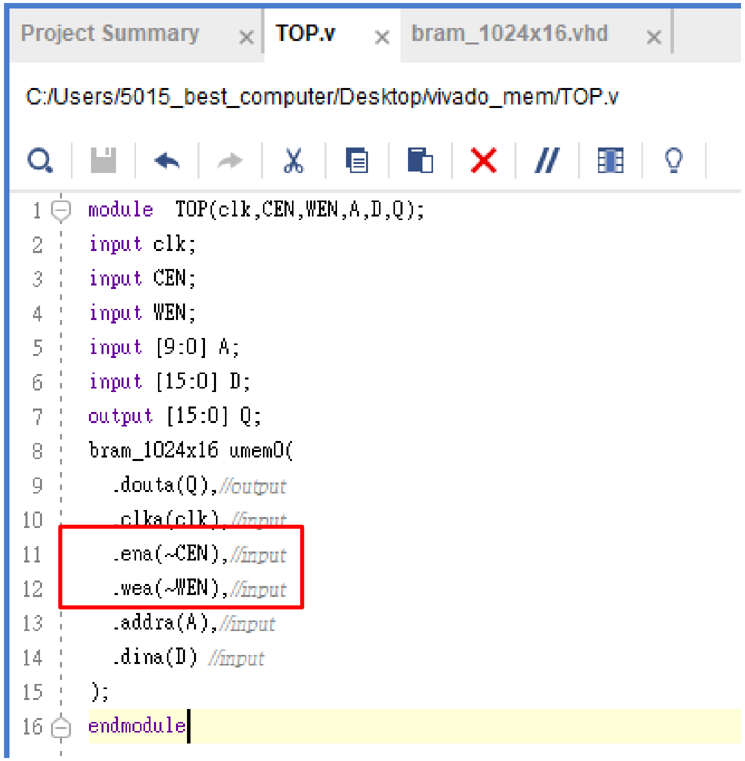
BRAM single port example (8/10)

- Step8.修改完成後可以看到TOP底下有bram了。
- CEN、WEN加反相(~)由於vivado與memory compiler的active是相反的，或是要改tb電路控制不用~也行。



Sources

- Design Sources (1)
  - TOP (TOP.v) (1)
    - umem0 : bram_1024x16 (bram_1024x16.xci) (1)
      - bram_1024x16(bram_1024x16_arch) (bram_1024x
        - U0 : blk_mem_gen_v8_4_2(xilinx)
- Constraints
- Simulation Sources (1)
  - sim_1 (1)
- Utility Sources

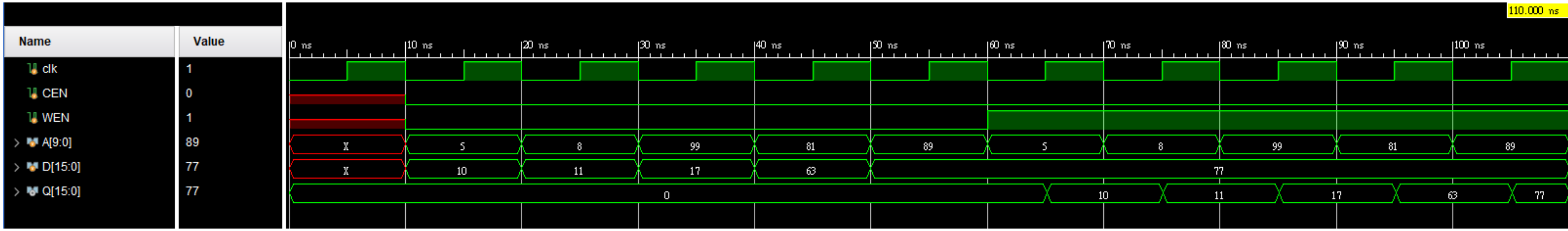


C:/Users/5015_best_computer/Desktop/vivado_mem/TOP.v

```
module  TOP(clk,CEN,WEN,A,D,Q);
input clk;
input CEN;
input WEN;
input [9:0] A;
input [15:0] D;
output [15:0] Q;
bram_1024x16 umem0(
   .douta(Q),//output
   .clka(clk),//input
   .ena(~CEN),//input
   .wea(~WEN),//input
   .addra(A),//input
   .dina(D) //input
);
endmodule
```

BRAM single port example (9/10)

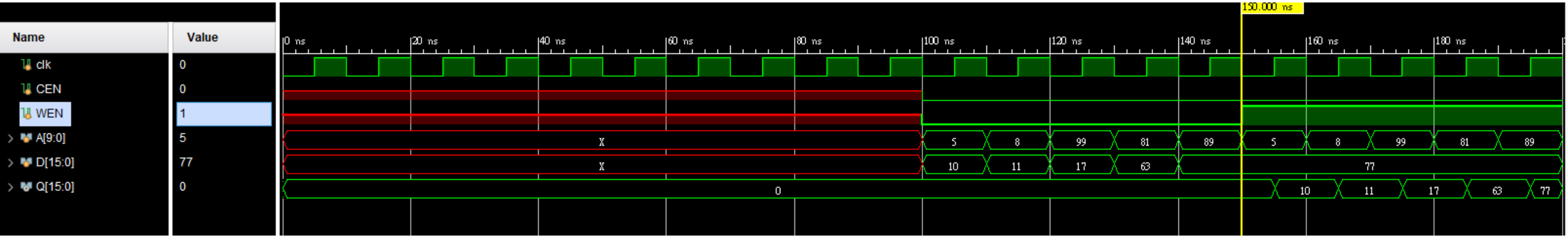
- Step9. Run Behav Simulation結果

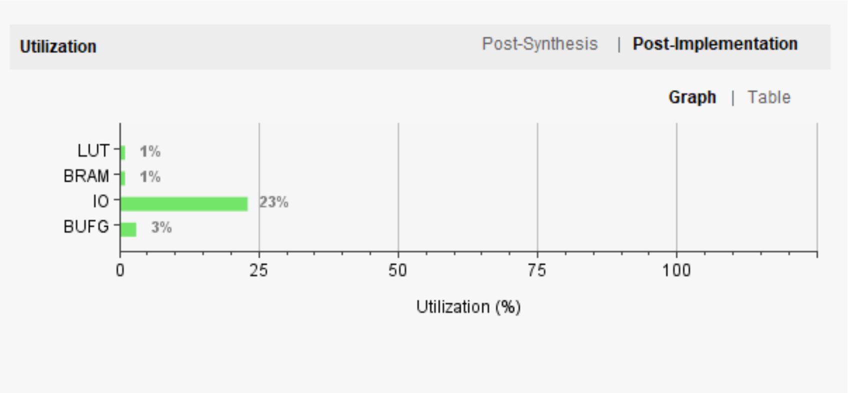


Name	Value
clk	1
CEN	0
WEN	1
A[9:0]	89
D[15:0]	77
Q[15:0]	77

BRAM single port example(10/10)

- Step10. 之後流程與之前vivado 的ppt一樣
- 跑Synthesis->設xdc->跑完Post-imp
- 出來的Post-imp波行與Uiltization


150.000 ns
Name
Value
clk 0
CEN 0
WEN 1
A[9:0] 5
D[15:0] 77
Q[15:0] 0
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
180 ns
X
5
8
99
81
89
5
8
99
81
89
X
10
11
17
63
77
0
10
11
17
63
77


Utilization
Post-Synthesis | Post-Implementation
Graph | Table
LUT
BRAM
IO
BUFG
1%
1%
23%
3%
0
25
50
75
100
Utilization (%)