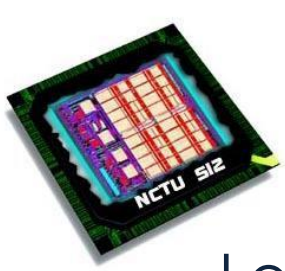
Introduction to Macros and SRAM

NYCU-EE IC LAB SPRING-2022



Lecturer: Yu-Wei Lu

Outline

- ✓ Section 1 – Macro
  - ✓ (Intellectual property, IP)
- ✓ Section 2 – Hard IP: Memory
  - ✓ Behavior
  - ✓ Usage

Outline

- ✓ **Section 1 – Macro**
  - ✓ **(Intellectual property, IP)**
- ✓ **Section 2 – Hard IP: Memory**
  - ✓ **Behavior**
  - ✓ **Usage**

Introduction to Intellectual Property

- ✓ **Intellectual Property (IP) core**
  - What: IP is a design of a logic function that specifies how the elements are interconnected
    // e.g. square root
  - Why: A designer can develop more quickly by applying IPs
  - How: IPs may be licensed to another party
  - **Soft macro(IP)**: Synthesizable RTL
    - Portable and Editable
    - Unpredictable in terms of performance, timing, area, or power
    - IP protection risks
  - **Firm macro(IP)**: Netlist format
    - Performance optimization under a specific fabrication technology
    - Need not synthesizing (sometimes it's time wasting)
  - **Hard macro(IP)**: Hardware (LEF, GDS2 file format)
    - Specifies the physical pathways and wiring (proved under specific tech.)
    - Moving, rotating, flipping freedom but can't touch the interior (APR)

- ✓ **Imagine that your design is for cellphone screen processing**
  - Assume the resolution is 1920*1080, 24 bits per pixel
  - 50M registers!!
- ✓ **Cellphone becomes large and power-consuming!**

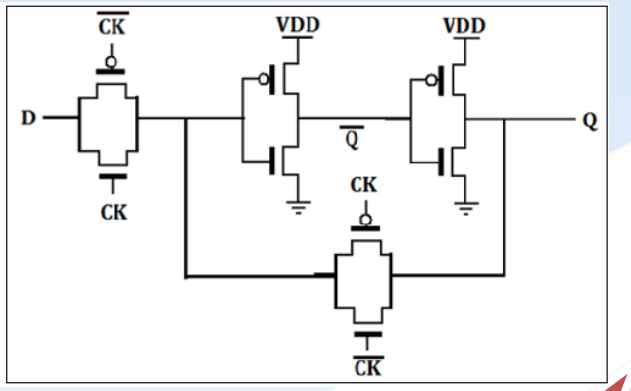
size

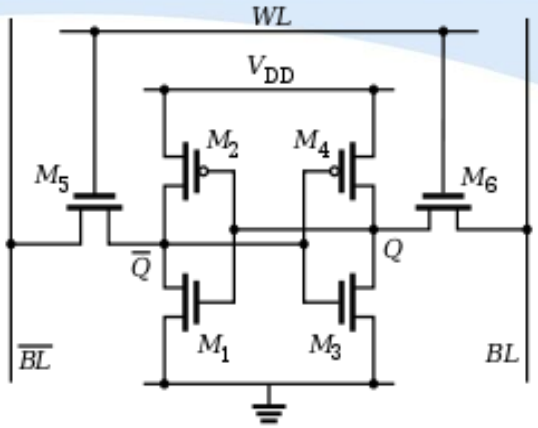
power

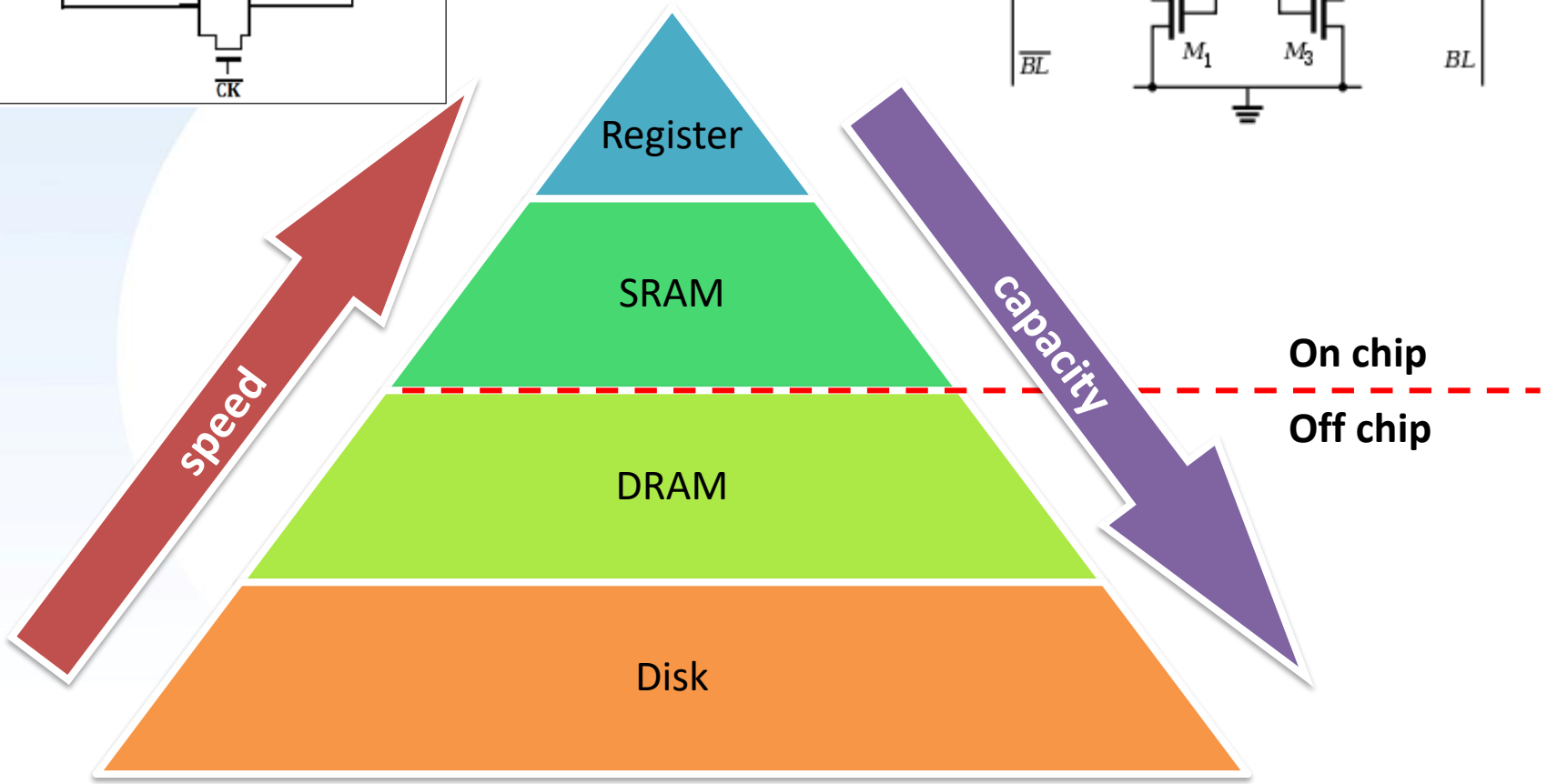
Outline

- ✓ Section 1 – Macro
  - ✓ (Intellectual property, IP)
- ✓ **Section 2 – Hard IP: Memory**
  - ✓ **Behavior**
  - ✓ **Usage**

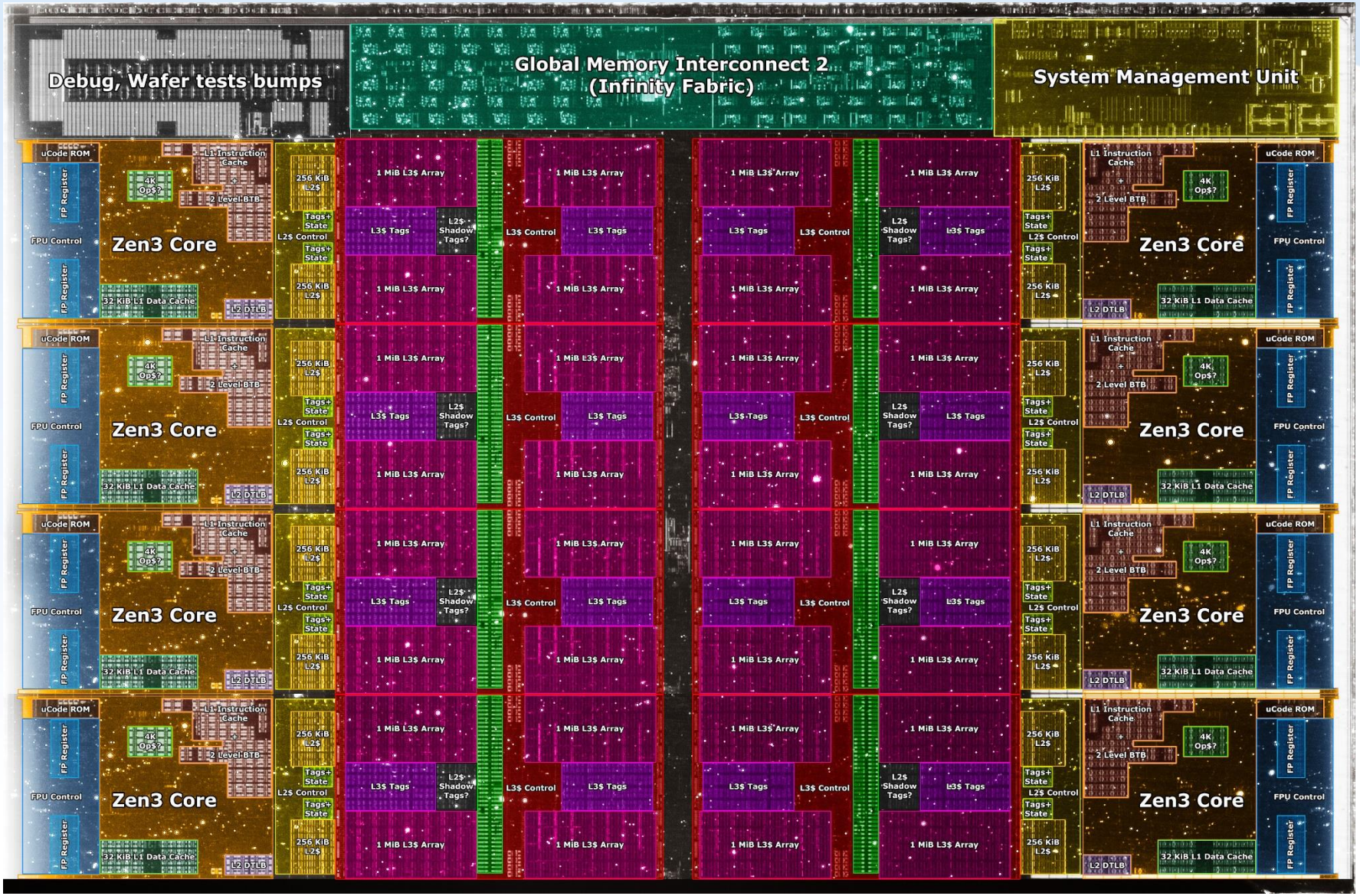
Memory Hierarchy







AMD Ryzen ZEN3



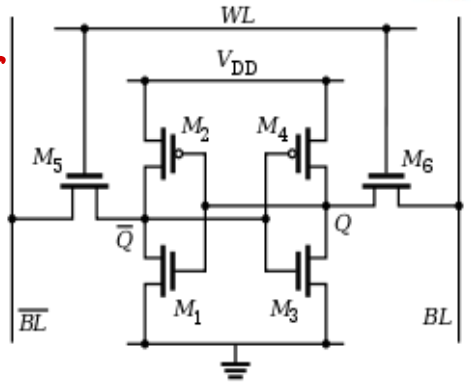
Memory

- ✓ **SRAM**
  - Read and Write Data only
  - Memory has less area than register
  - Memory is slower than register
  - Only one address can be accessed in the same time (single port SRAM vs. dual port)

DRAM ⇒ Delay longer

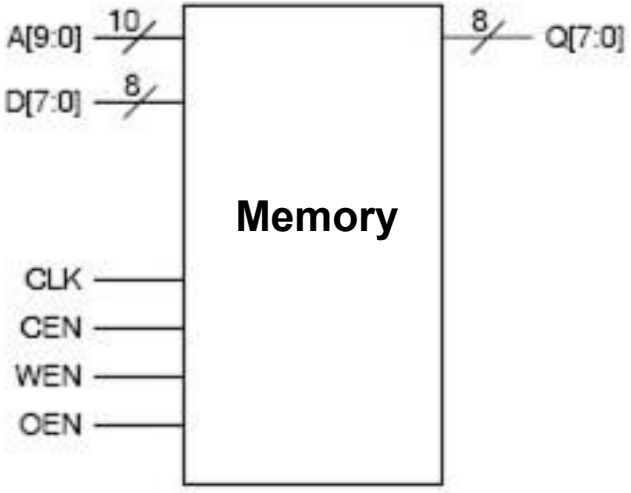
SRAM ⇒ larger

6T SRAM



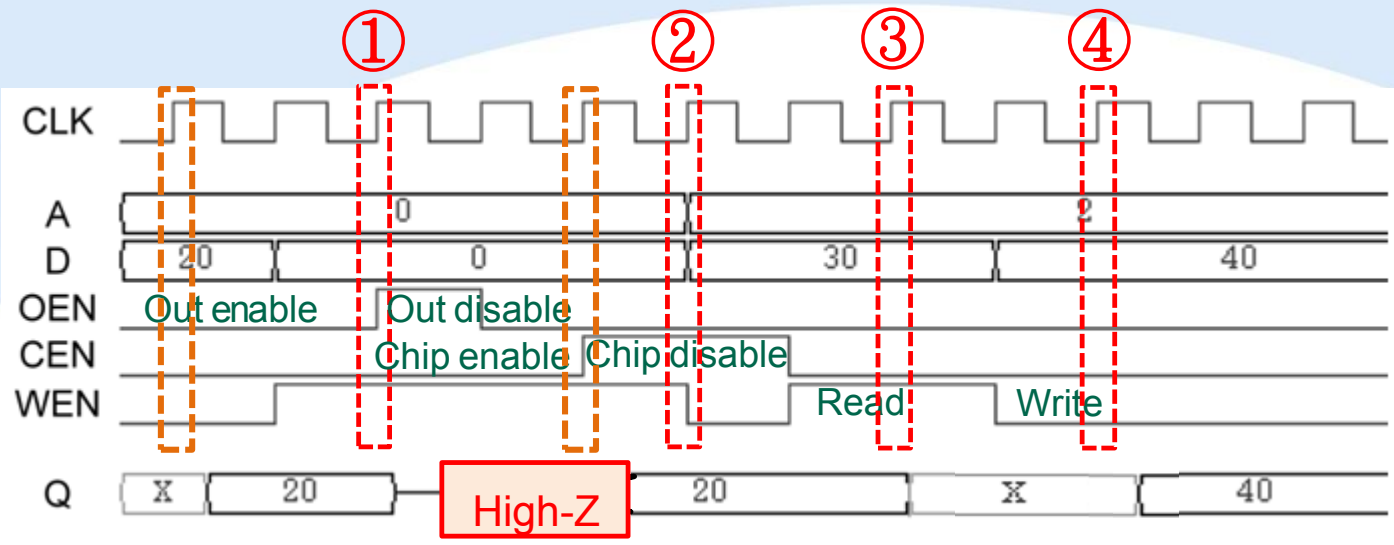
- ✓ **Single port SRAM I/O Description**

Pin	Description
A[9:0]	Address(A[0]=LSB)
D[7:0]	Data input(D[0]=LSB)
CLK	Clock input
CEN	Chip Enable Negative
WEN	Write Enable Negative
OEN	Output Enable Negative
Q[7:0]	Data Output(Q[0]=LSB)



SRAM Logic Table

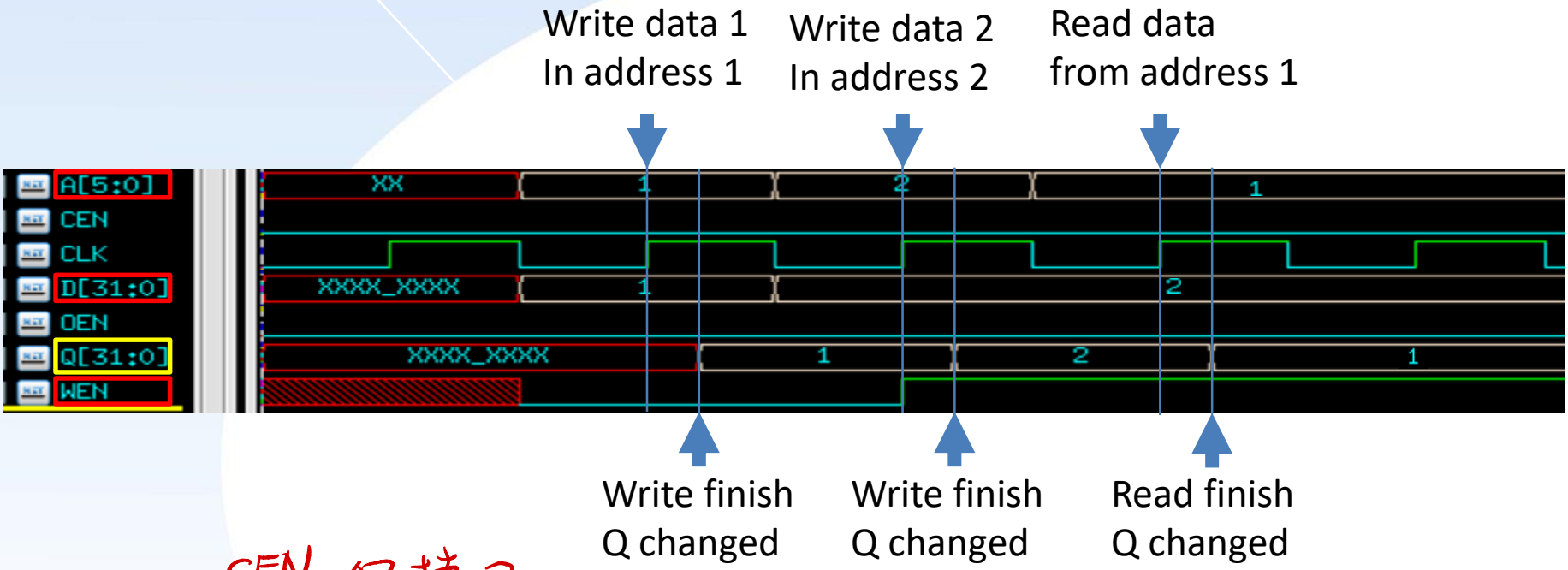
- OEN is a tri-state buffer
- Considering CLK skew, Enable Chip at least one cycle before use



SRAM Logic Table

	CEN	WEN	OEN	Data Out	Mode	Function
①	X	X	H	Z	High-Z	The data output bus Q[n-1:0] is placed in a high impedance state. Other memory operations are unaffected.
②	H	X	L	Last Data	Standby	Address inputs are disabled; data stored in the memory is retained, but the memory cannot be accessed for new reads or writes. Data outputs remain stable.
③	L	H	L	SRAM Data	Read	Data on the data output bus Q[n-1:0] is read from the memory location specified on the address bus A[m-1:0].
④	L	L	L	Data In	Write	Data on the data input bus D[n-1:0] is written to the memory location specified on the address bus A[m-1:0], and driven through to the data output bus Q[n-1:0].

Signal example



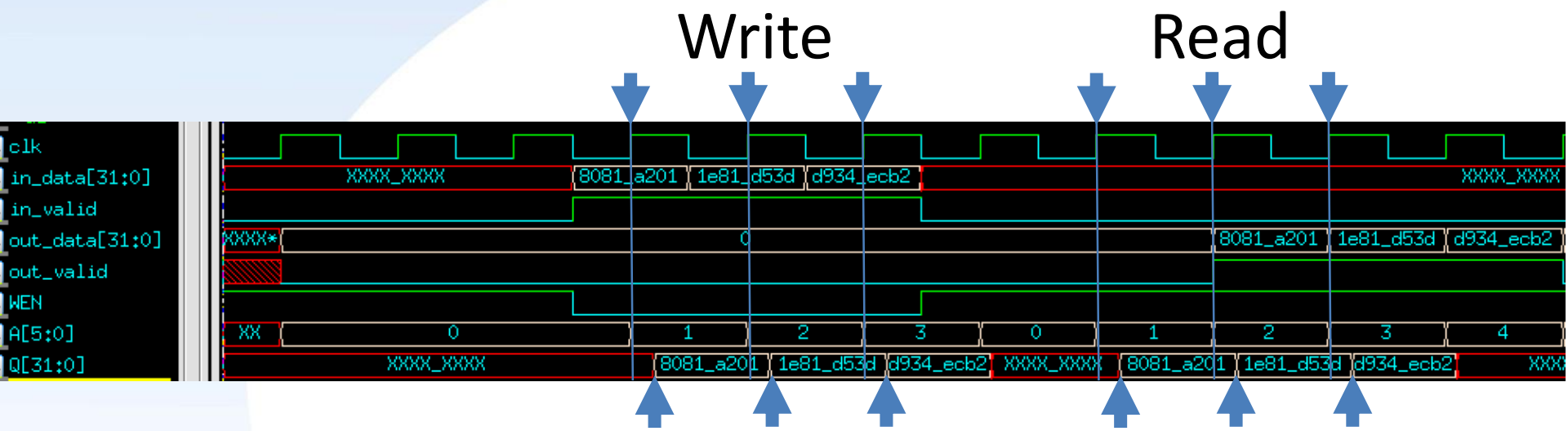
CEN
OEN 保持0

用WEN控制

WEN (write enable negative):

- 0 -> write
- 1 -> read

Appendix-Write and read in order



Appendix-Write and read in order

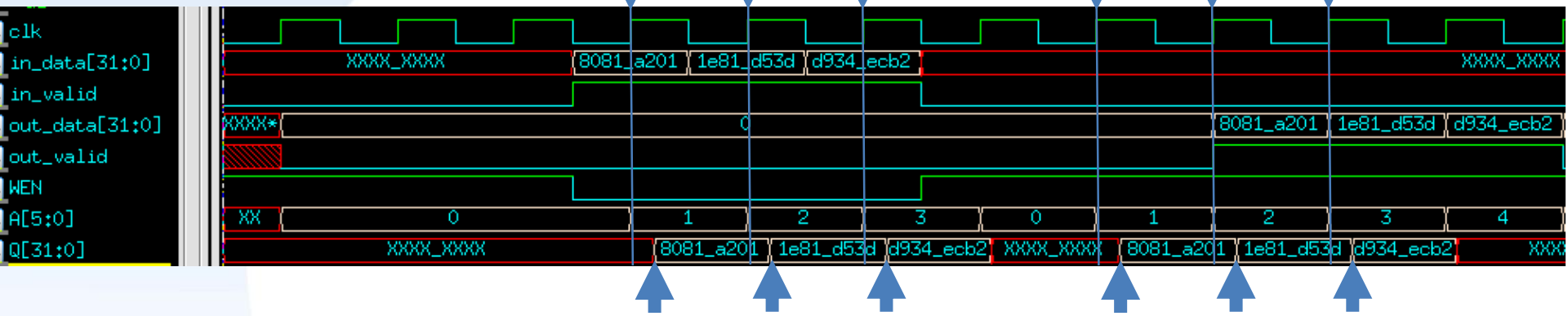
A	D
0	X
1	X
2	X
3	X

A	D
0	8
1	X
2	X
3	X

A	D
0	8
1	1
2	X
3	X

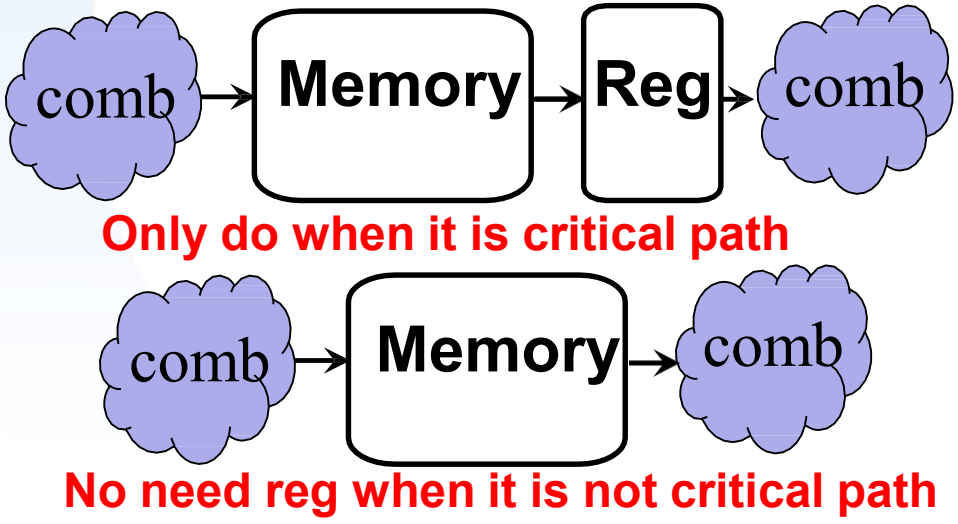
A	D
0	8
1	1
2	d
3	X

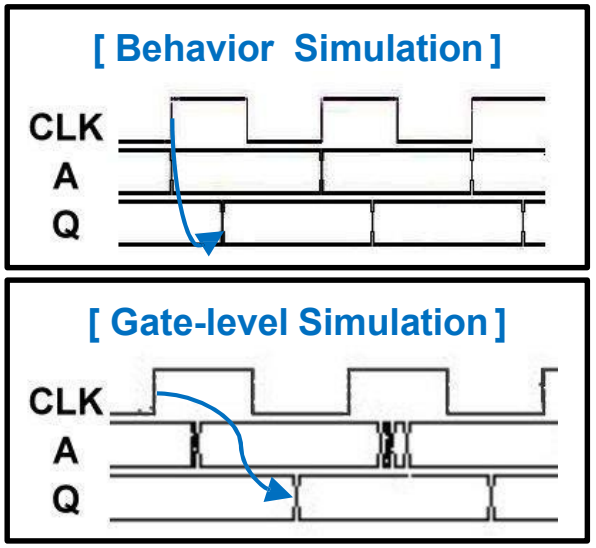
Write Read



Design Tips

- ✓ **To avoid critical path causing timing violation**
  - Add registers after the hard macro
  - Use enable signal to control output register to avoid reading unknown value
- ✓ **If a memory macro is used in your design, the timescale should be set according to the timescale specified by memory file**
- ✓ **Be aware of features and characteristics of hard macro before you use it in your design**



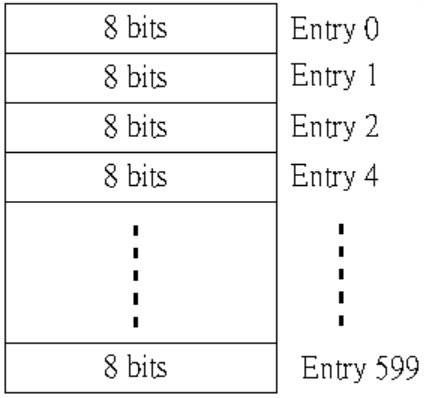


Memory generation example

Example :

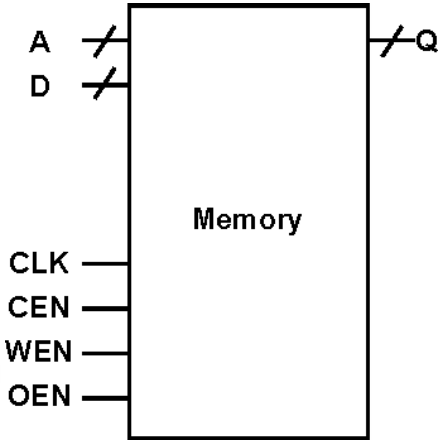
Number of Words : 600

Number of Bits : 8



1. How many bits of input pins are needed?

2. How many bits of address are needed?



Memory generation example

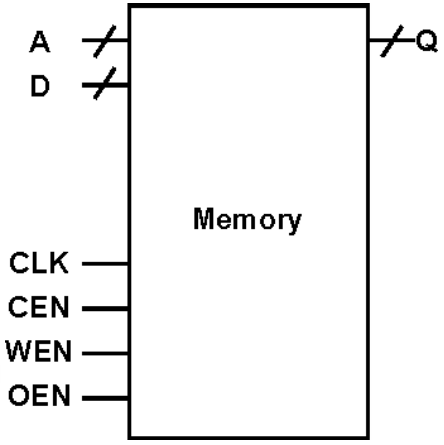
Example :

Number of Words : 600

Number of Bits : 8

1. How many bits of input pins are needed?
2. How many bits of address are needed?

1. How many bits of input pins are needed?
2. How many bits of address are needed?



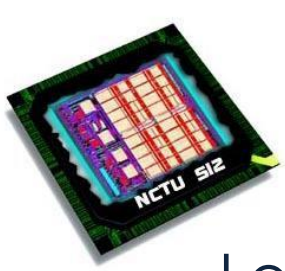
- **Answer :**
  - **D [7:0]**
  - **Q [7:0]**
  - **A [9:0]**

→ $log_2 600 = 9.2288186 \approx 10$

• Answer :
- D [7:0]
- Q [7:0]
- A [9:0]
→ $log_2 600 = 9.2288186 \approx 10$

Memory Compiler

NYCU-EE IC LAB SPRING-2022



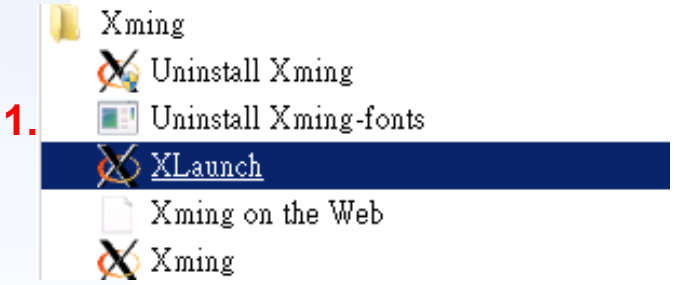
Lecturer: Yu-Wei Lu

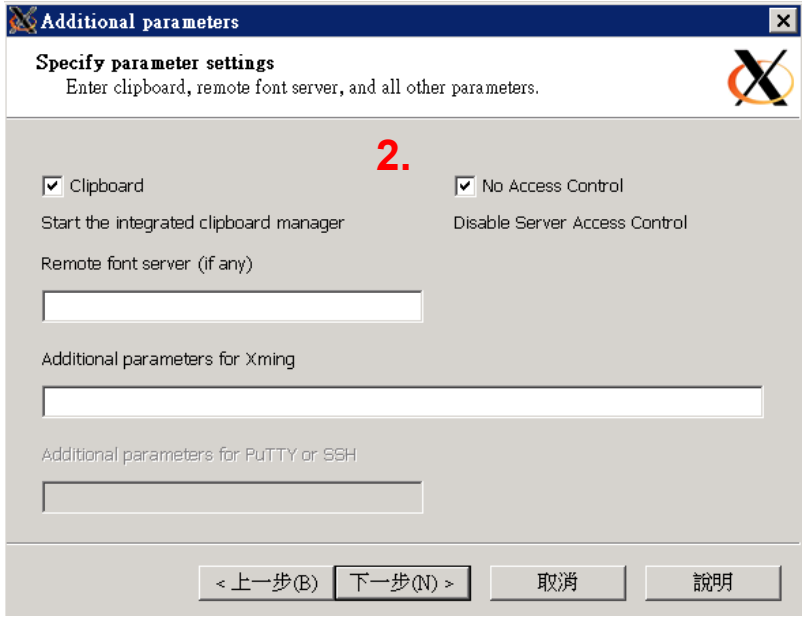
Outline

- ✓ **Section 1 – GUI**
- ✓ **Section 2 – Script**

Memory Compiler GUI steps

- ✓ **Step 1. Execute Xlaunch**
  - https://sourceforge.net/projects/xming/
- ✓ **Step 2. Check No Access Control**
- ✓ **Step 3. Press Next or Yes for all other pages**





Memory Compiler GUI steps

- **Step 4. Log in linux01.ee.nctu.edu.tw**

```
**************************************************************************
* CAD Tools available on this machine:                                   *
**************************************************************************
 Cadence:  IES_1210006 MMSIM_61 EDI_1013005
 Synopsys: DC0809 PT0812 PP0606 HSPICE0909
 Novas:    Verdi2006_07 Laker32
 Mentor:   Calibre_1033726
 Syntest:  Syntest & TurboScan
 Mathworks:  Matlab2009a
 Xilinx: ISE

linux01 [iclab/iclabta05]%
```

- **Step 5. Connect to ee08**

  Using ssh to connect the server, which is

  %ssh mem@ee08.ee.nctu.edu.tw **pw: mem**

```
linux01 [iclab/iclabta05]% ssh mem@ee08.ee.nctu.edu.tw
mem@ee08.ee.nctu.edu.tw's password:
Last login: Wed Apr 18 2018 00:24:51 +0800 from linux01
Sun Microsystems Inc.   SunOS 5.8       Generic Patch   February 2004
No mail.
Sun Microsystems Inc.   SunOS 5.8       Generic Patch   February 2004
$
```

  - You could also directly log in with username "mem" at ee08, just like log in your account in other server

Memory Compiler GUI steps

- ✓ **Step 6: create your own directory**

  $ mkdir your_own_name (ex: iclabxxx)

  $ cd your_own_name (ex: iclabxxx)

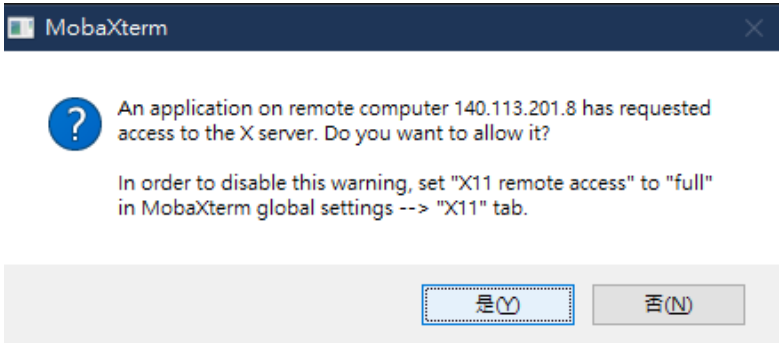
- ✓ **Step 7. run the following commands**

  $ setenv DISPLAY 140.113.x.x:0

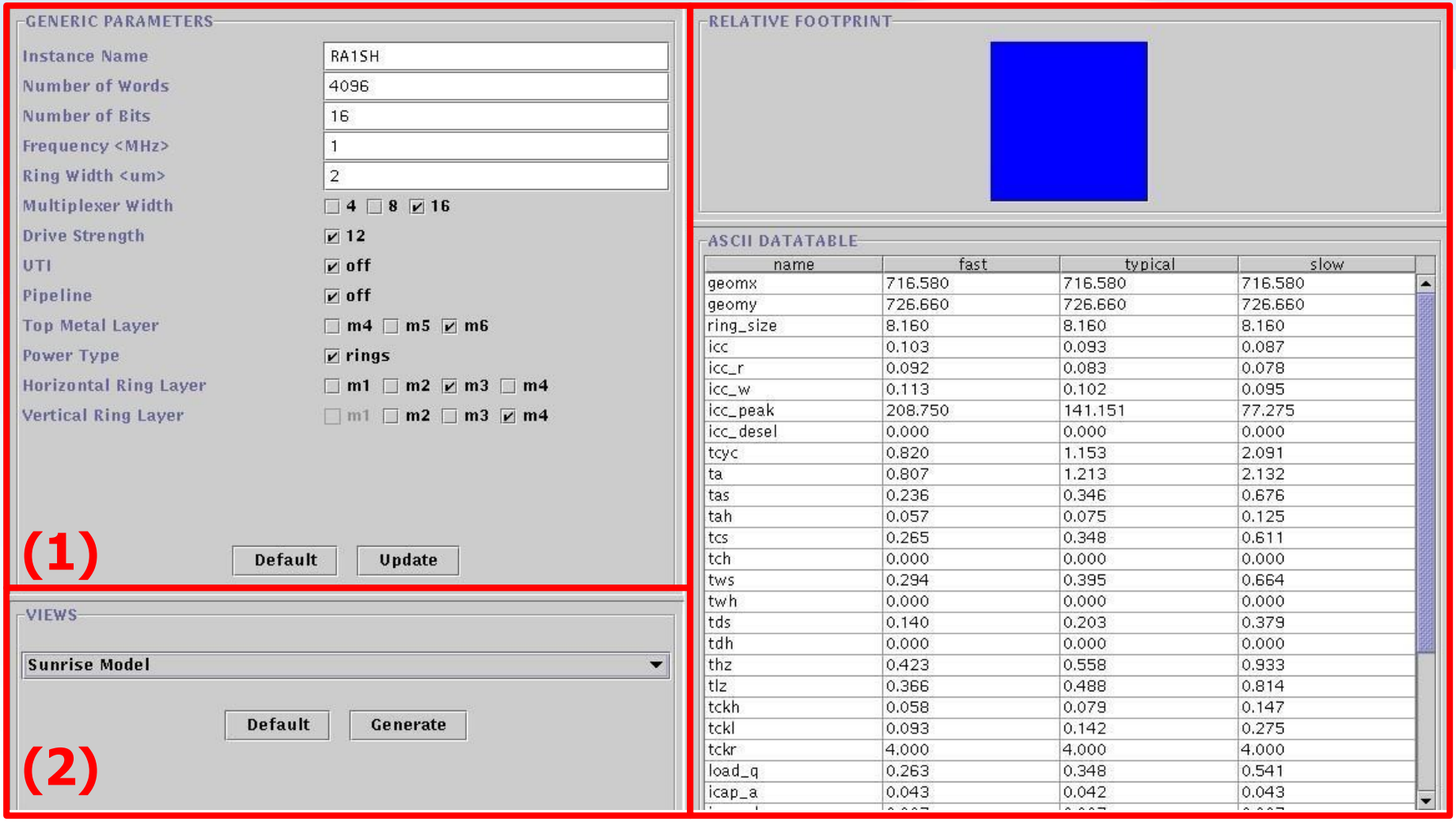
  $ /RAID2/EDA/memory/CBDK018_UMC_Artisan/orig_lib/aci/ra1sh_1/bin/ra1sh

- ✓ **IP must be 140.113.x.x**

- ✓ **Press Y to allow requested access to the X server**



Memory Compiler Interface



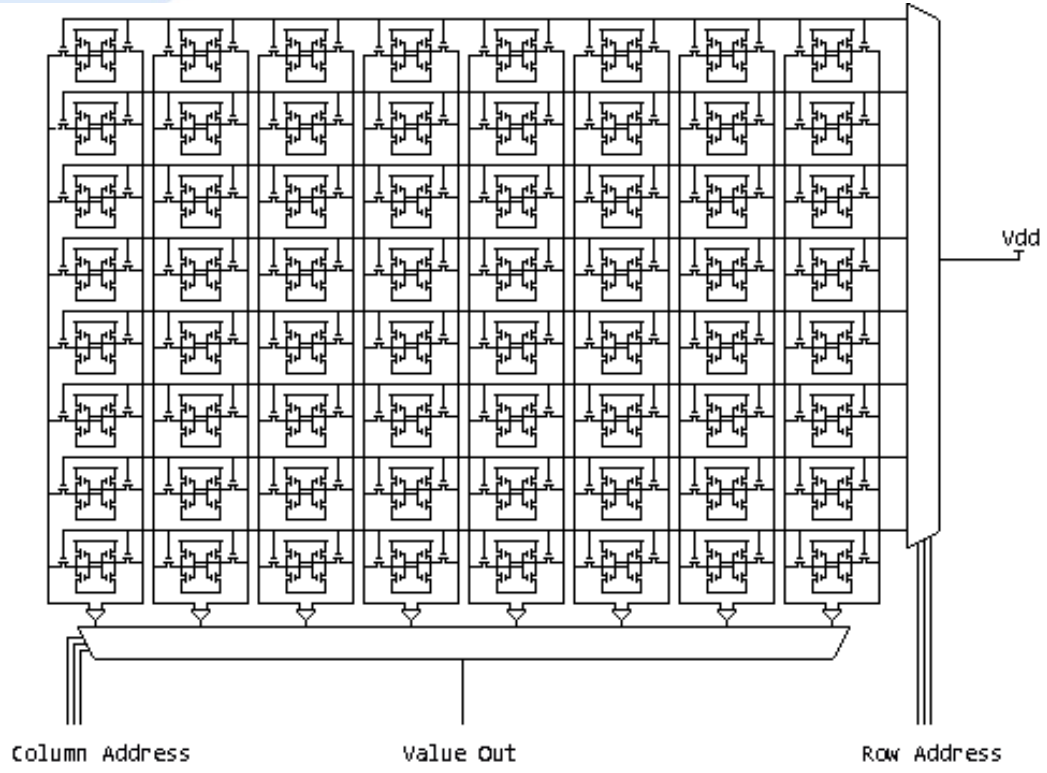
name	fast	typical	slow
geomx	716.580	716.580	716.580
geomy	726.660	726.660	726.660
ring_size	8.160	8.160	8.160
icc	0.103	0.093	0.087
icc_r	0.092	0.083	0.078
icc_w	0.113	0.102	0.095
icc_peak	208.750	141.151	77.275
icc_desel	0.000	0.000	0.000
tcyc	0.820	1.153	2.091
ta	0.807	1.213	2.132
tas	0.236	0.346	0.676
tah	0.057	0.075	0.125
tcs	0.265	0.348	0.611
tch	0.000	0.000	0.000
tws	0.294	0.395	0.664
twh	0.000	0.000	0.000
tds	0.140	0.203	0.379
tdh	0.000	0.000	0.000
thz	0.423	0.558	0.933
tlz	0.366	0.488	0.814
tckh	0.058	0.079	0.147
tckl	0.093	0.142	0.275
tckr	4.000	4.000	4.000
load_q	0.263	0.348	0.541
icap_a	0.043	0.042	0.043

Memory Compiler Parameter

- **Instance Name : memory name**
- **Number of Words : number of entry for the designed memory**
- **Number of Bits : number of bits for every entry**
- **Frequency <MHz> : memory working frequency**
- Ring Width : power line width
- **Multiplexer Width : 4-to-1, 8-to-1, 16-to-1 multiplexer**
- Top Metal Layer : the highest level of metal can be used in memory

Memory Compiler

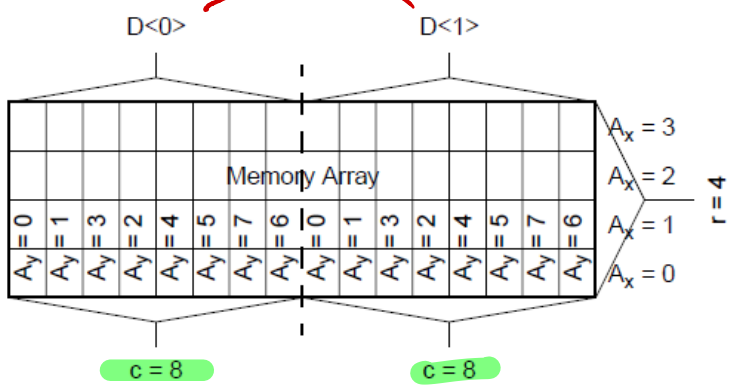
Memory Architectures



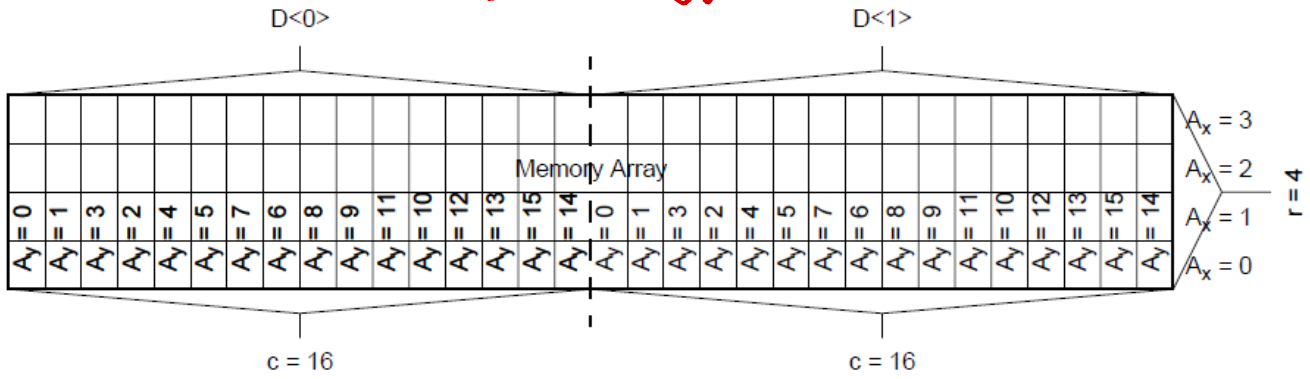
Memory Compiler

- **Example :**

32 words, 2bit , Multiplexer Width = 8



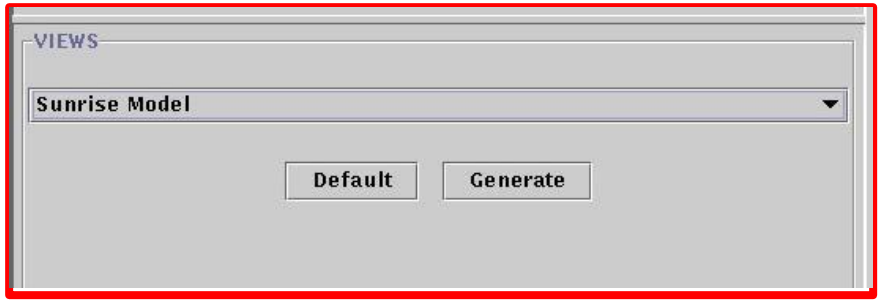
64 words, 2bit , Multiplexer Width = 16



- **Hint: change multiplexer width to make footprint close to square.**
  - $(bit \times Mux\ Width) \approx (Words \div Mux\ Width)$
  - $\Rightarrow Mux\ Width^2 \approx Words \div bit$

Memory Compiler

Spec.	**PostScript Datasheet :** data sheet (*.ps) (use ps2pdf for .pdf) **ASCII datatable :** parameter table (*.dat)
For designer	**Verilog model :** behavior model (*.v) (can't be synthesized) **Synopsys model :** library for synthesis & APR (*.lib)
For APR	**LVS Netlist :** used for LVS **GDSII Layout :** layout file



Outline

✓ Section 1 – GUI

✓ **Section 2 – Script**

SRAM generation in this course

- **Step 1. Log in linux??.ee.nctu.edu.tw**

```
*********************************************************************
* CAD Tools available on this machine:                              *
*********************************************************************
 Cadence:  INCISIV_13.10.005 MMSIM_15.10.801
 Synopsys: DC_2015.06 PT_2013.12 hspice_2015.06
 Novas:    Verdi_2015.09 Laker_2015.03
 Mentor:   Calibre_2008.2_22.20
 Cadence:  Jasper Gold_2021.03
 Synopsys:  MetaWare Development Toolkit
 Synopsys:  Tetra Max
 Cadence:  Innovus_17.11
*********************************************************************
linux01 [iclab/iclabta02]%
```

- **Step 2. Connect to ee08**

  Using ssh to connect the server, which is

  %ssh mem@ee08.ee.nctu.edu.tw **pw: mem**

```
linux01 [iclab/iclabta02]% ssh mem@ee08.ee.nctu.edu.tw
mem@ee08.ee.nctu.edu.tw's password:
Last login: Fri Oct 01 2021 15:18:59 +0800 from si2pc19.EE.NCTU.
```

SRAM generation in this course

- **Step 3. Copy the directory /template and name on your own**

  % cp -r template your_own_name (ex: iclabxxx)

```
$ ls
template
$ cp -r template iclab777
$ ls
iclab777  template
$
```

- **Step 4. Generate the memory you need in your directory**

  % cd your_own_name (ex:iclabxxx)

  % ./01_mem_gen.sh RAISH 256 33 16 200

```
$ ls
iclab777  template
$
$
$ cd iclab777
$ ls
01_mem_gen.sh               09_clean                RA1SH.v                 RA1SH_slow_syn.lib
02_lib_gen_syntax_match.sh  RA1SH.db                RA1SH_backup.v          RA1SH_typical_syn.lib
08_delete_mem               RA1SH.ps                RA1SH_fast_syn.lib      lc_shell.tcl
$ ./01_mem_gen.sh
give 3 inputs under the order:
    number of words
    number of bits
    mux type(4|8|16)

$ ./01_mem_gen.sh 256 33 16
```

SRAM generation in this course

- **Step 5. Copy the directory back to your account**

  $scp -r <your_mem_dir> <your_account>@linux01.ee.nctu.edu.tw:.

```
$ ls
iclab777  template
$ scp -r iclab777 iclabta05@linux01.ee.nctu.edu.tw:.
iclabta05@linux01.ee.nctu.edu.tw's password:
iclabta05@linux01.ee.nctu.edu.tw's password:
02_lib_gen_syntax_match.sh                |  267B |  267B/s | TOC: 00:00:01 | 100%
08_delete_mem                             |   24B |   24B/s | TOC: 00:00:01 | 100%
09_clean                                  |   11B |   11B/s | TOC: 00:00:01 | 100%
lc_shell.tcl                              |   79B |   79B/s | TOC: 00:00:01 | 100%
RA1SH.db                                  |  20kB |  20kB/s | TOC: 00:00:01 | 100%
RA1SH.ps                                  | 156kB | 156kB/s | TOC: 00:00:01 | 100%
RA1SH.v                                   |  19kB |  19kB/s | TOC: 00:00:01 | 100%
RA1SH_backup.v                            | 9.6kB | 9.6kB/s | TOC: 00:00:01 | 100%
RA1SH_fast_syn.lib                        |  18kB |  18kB/s | TOC: 00:00:01 | 100%
RA1SH_typical_syn.lib                     |  18kB |  18kB/s | TOC: 00:00:01 | 100%
RA1SH_slow_syn.lib                        |  18kB |  18kB/s | TOC: 00:00:01 | 100%
01_mem_gen.sh                             |  296B |  296B/s | TOC: 00:00:01 | 100%
$ exit
Connection to ee08.ee.nctu.edu.tw closed.
linux01 [iclab/iclabta05]% ls
Desktop    FP        Lab02        Lab03_DEMO       Lab04_Demo  Pictures   Videos
Documents  Lab002    Lab02_DEMO   Lab03_DEMOOO     Lab05       Public     iclab777
```

SRAM generation in this course

- ✓ **Step 6. Generate db file**
- ✓ **Step 7. Match the syntax of v file**

Use library compiler to generate .db files from .lib files

➢ **Step 6. Generate (*.db) from (*.lib) for (xx.tcl) usage**

./02 ...

– Invoke Synopsys
  % **lc_shell**

– Once inside lc_shell, execute the following Synopsys commands
  lc_shell> **read_lib** xxx**.lib**
  lc_shell> **write_lib -format db USERLIB -output** xxx**.db**
  ◆ Note: Name of (*.lib) and (*.db) must be the same.

– Exit lc_shell
  lc_shell> **exit**

✓ **After generating the Synopsys model (*.db), one can generate SDF**

Match the syntax of *.v file to SDF

➢ Step 7. Match the syntax

- ✓ **Specify the setup time and hold time by replacing**

  $setuphold(posedge CLK &&& re_data_flag,D[9], 1.000, 0.500, NOT_D9);

  to

  $setuphold(posedge CLK &&& re_data_flag,**posedge** D[9], 1.000, 0.500, NOT_D9);
  $setuphold(posedge CLK &&& re_data_flag,**negedge** D[9], 1.000, 0.500, NOT_D9);

- ✓ **Specify the delay of io paths by replacing**

  (CLK => Q[0])=(1.000, 1.000, 0.500, 1.000, 0.500, 1.000);

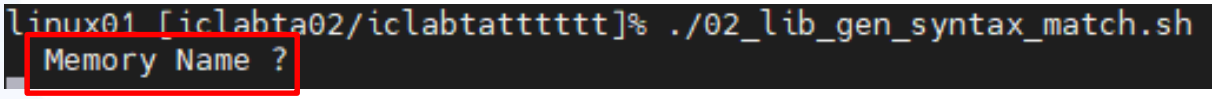
  to

  (posedge CLK => (Q[0]:1'bx))=(1.000, 1.000, 0.500, 1.000, 0.500, 1.000);

- ◆ Note
  - Input part includes CEN, WEN, A, and D
  - Output part includes CLK to Q, and OEN to Q

Step6, 7 command with a single command

- ✓ **Step 6, 7 can be done with a single command.**
- ✓ **Take care not to perform the same action twice.**
  (ex: manually changed an run the command again)
- ✓ **If you do wrong, you should redo from generating the memory.**
- ✓ **Command: ./02_lib_gen_syntax_match.sh**



- ✓ **With this command, the db file will be generated and .v file will be changed automatically**

Move files

- ✓ **Step 8. After you get .db file and .v file, put them to Exercise/04_MEM folder.**
- ✓ **Step 9. Edit the file_list.f in /01_RTL/ folder.**

For example：

```
../04_MEM/RA1SH.v
```

- ✓ **Then you can use them to run behavior simulation and synthesis.**

Remind!

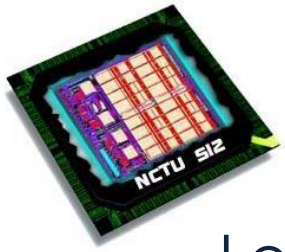
- ✓ **When using IP, information in lib file belong to certain module name, so modifying module name in v file is forbidden.**

```
module RA1SH1 (
    Q,
    CLK,
    CEN,
    WEN,
    A,
    D,
    OEN
);
    parameter           BITS = 38;
    parameter           word_depth = 1056;
    parameter           addr_width = 11;
    parameter           wordx = {BITS{1'bx}};
    parameter           addrx = {addr_width{1'bx}};
```

can not modify

JasperGold Superlint

NYCU-EE IC LAB SPRING-2022



Lecturer: Yu-Wei Lu

Outline

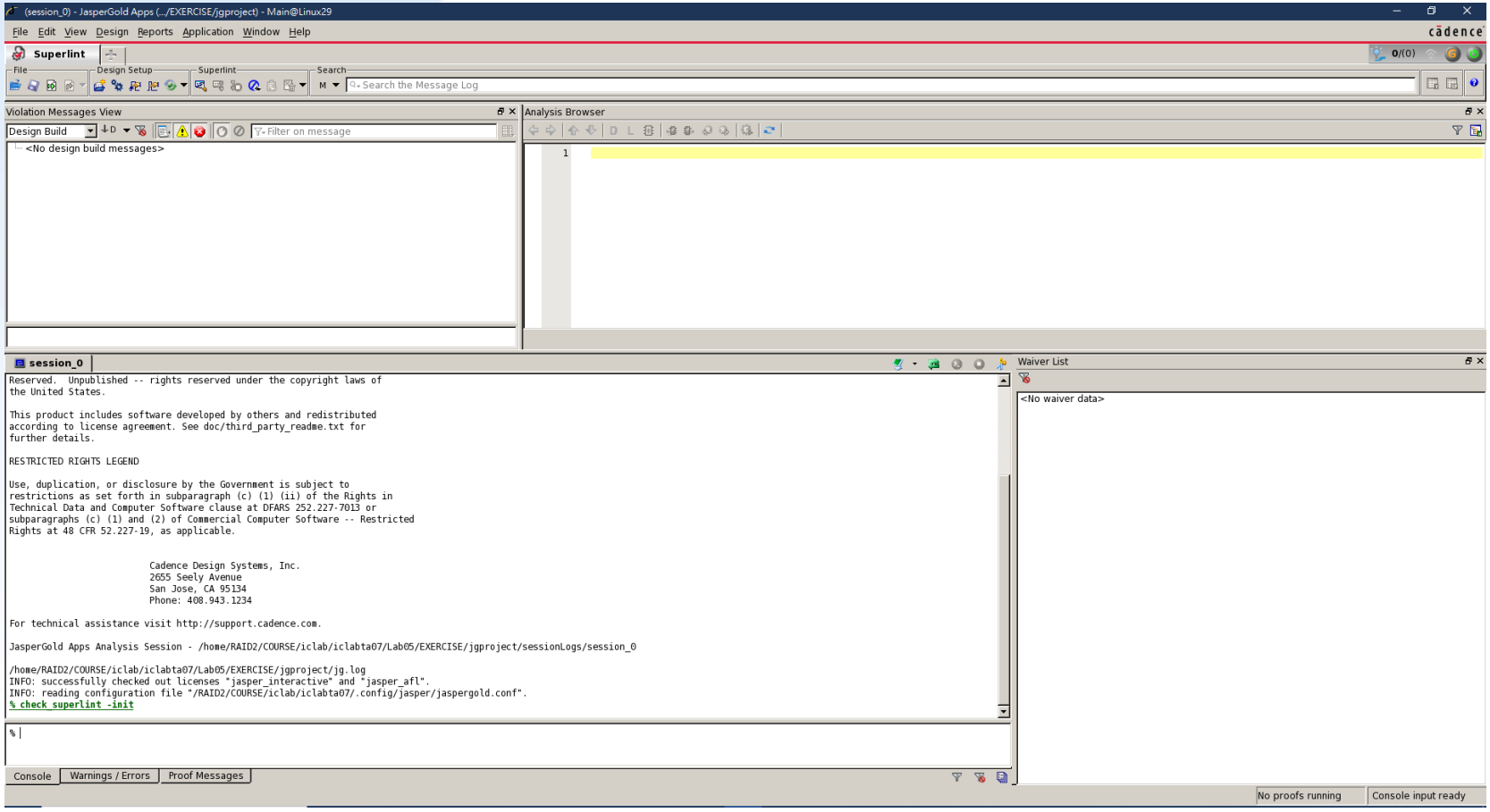
- ✓ Overview
- ✓ Choose Configure Checks
- ✓ Import Design File
- ✓ Setup the Clock and Reset
- ✓ Extract and Prove Superlint Checks

Overview

- ✓ **Superlint combines traditional RTL linting and formal analysis, deriving rich property-based functional checks from the RTL automatically.**
- ✓ **Superlint includes comprehensive lint and DFT checks.**
- ✓ **Two modes of superlint**
  - Command line mode (batch mode)
  - Graphic User Interface(GUI) mode: **user-friendly!!**

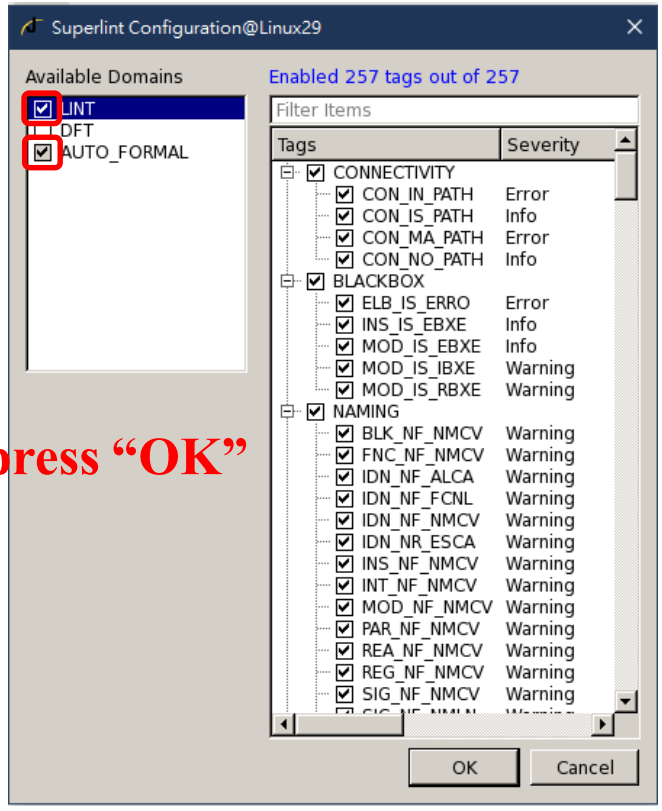
Overview (cont.)

- **Invoke JasperGold Superlint:**
  - By command % **jg –superlint &**



Choose Configure Checks

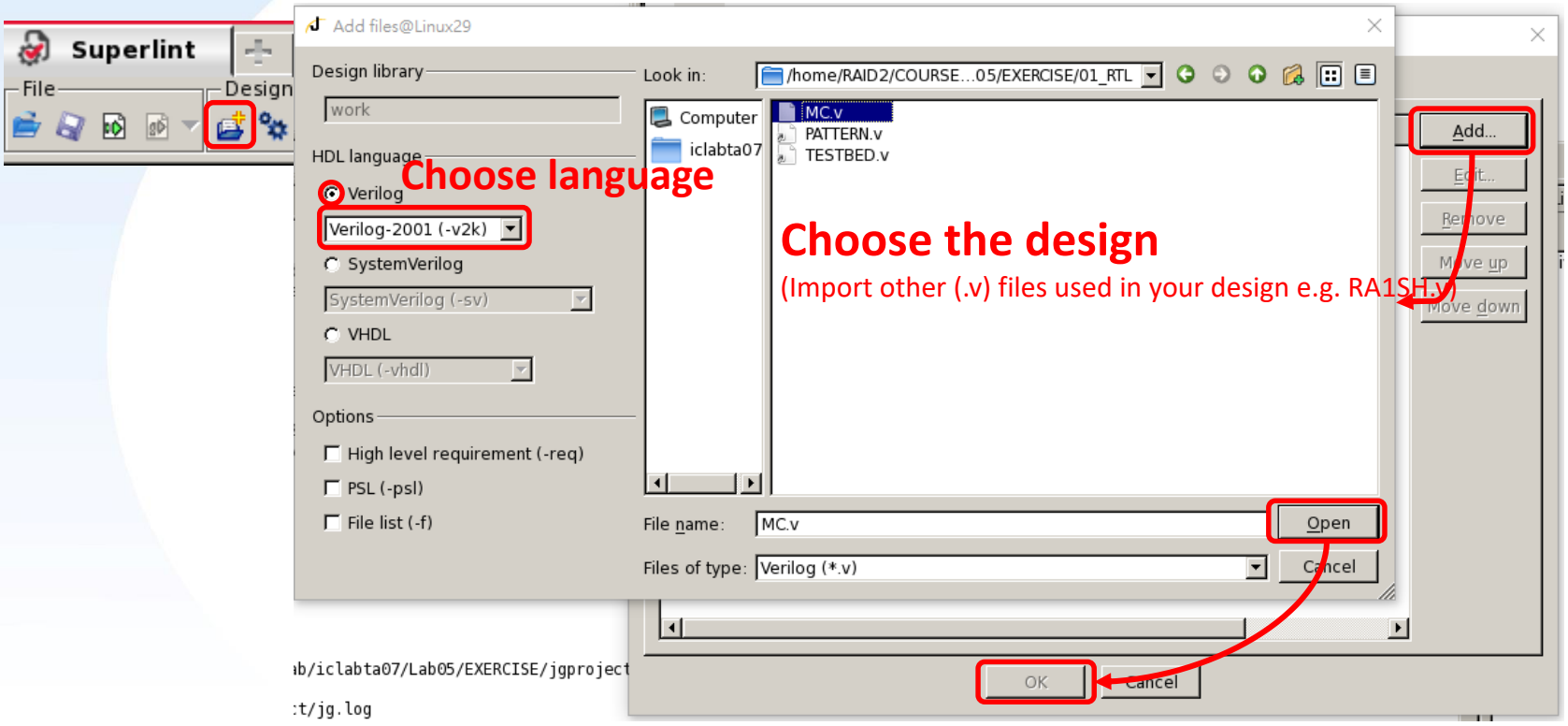
- **Configure Checks to Run**
  - Using the ***Application -> Configure Superlint Checks***
  - Or press the button



Choose the configuration, then press "OK"

Import Design File

- ✓ **Analyze the Design**
  - Using the ***Design -> Analyze RTL***
  - Or press the button



✓ If using DesignWare IP

– 「–bbox_m XXXX」

```
INFO: reading configuration file "/RAID2/COURSE/iclab/iclabta02/.config/jasper/jaspergold.co
% check_superlint -init
% analyze -v2k {/home/RAID2/COURSE/iclab/iclabta02/Lab05_2021fall/EXERCISE/01_RTL/TMIP.v} ;
  analyze -v2k {/home/RAID2/COURSE/iclab/iclabta02/Lab05_2021fall/EXERCISE/04_MEM/RA1SH256VER2.v} ;
```

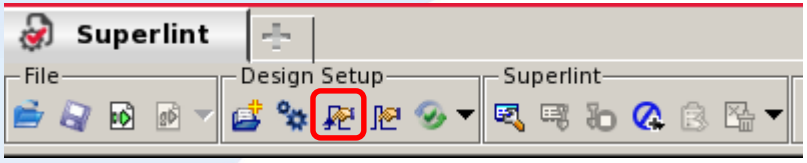
```
INFO: reading configuration file "/RAID2/COURSE/iclab/iclabta02/.config/jasper/jaspergold.co
% check_superlint -init
% analyze -v2k {/home/RAID2/COURSE/iclab/iclabta02/Lab05_2021fall/EXERCISE/01_RTL/TMIP.v} ;
  analyze -v2k {/home/RAID2/COURSE/iclab/iclabta02/Lab05_2021fall/EXERCISE/04_MEM/RA1SH256VER2.v} -bbox_m DE_minmax;
```

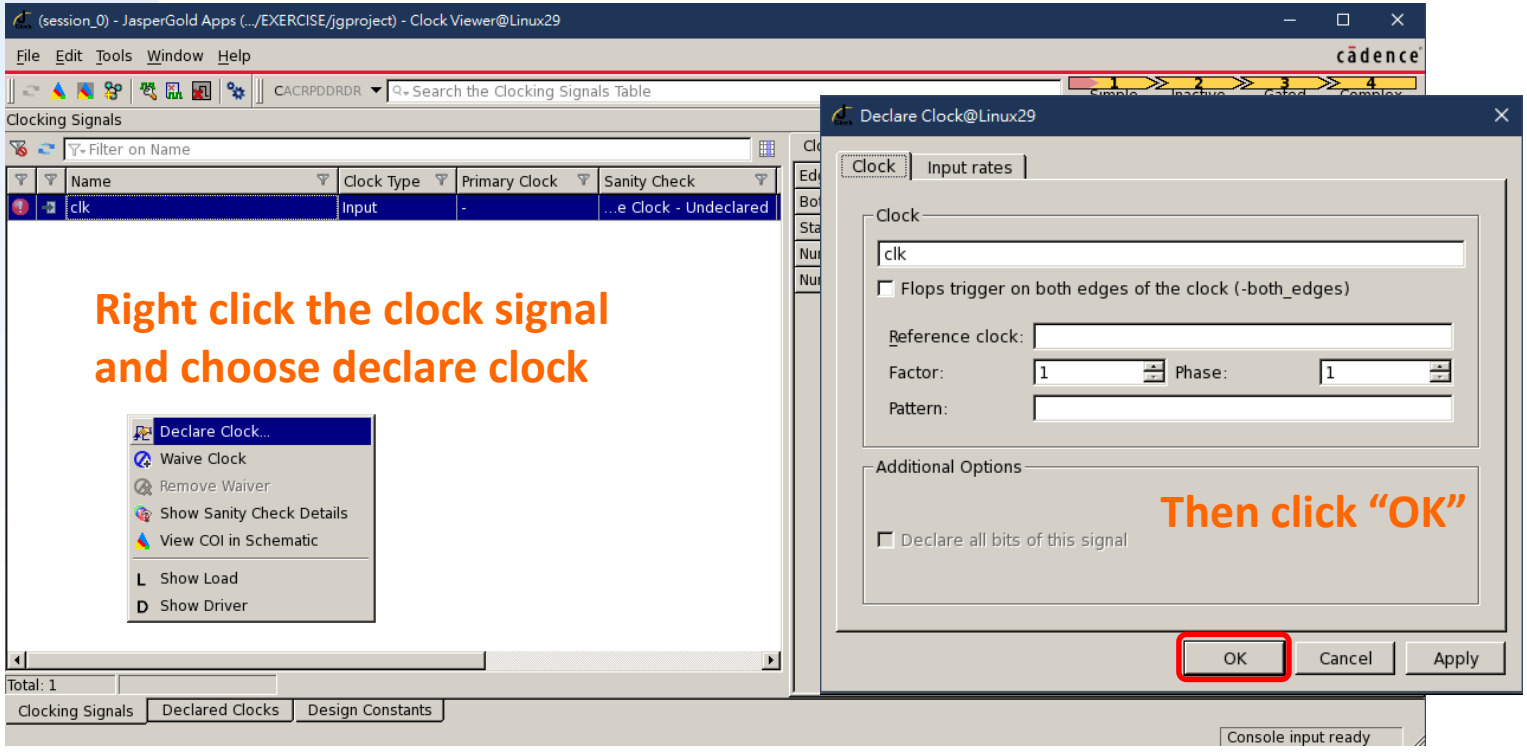
✓ Elaborate

```
% elaborate
```

Setup the clock and Reset

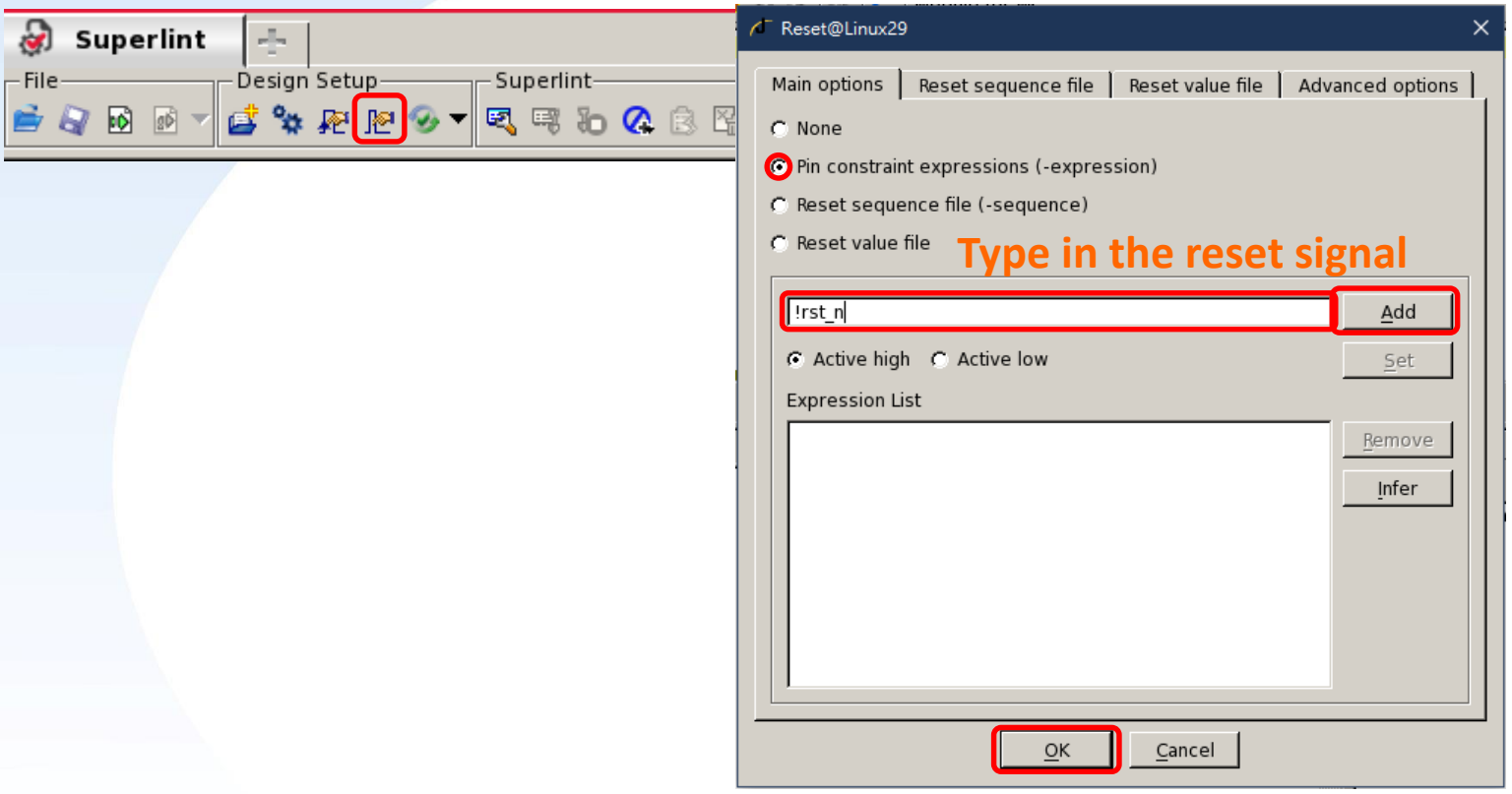
- ✓ **Setup the Clock**
  - press the button





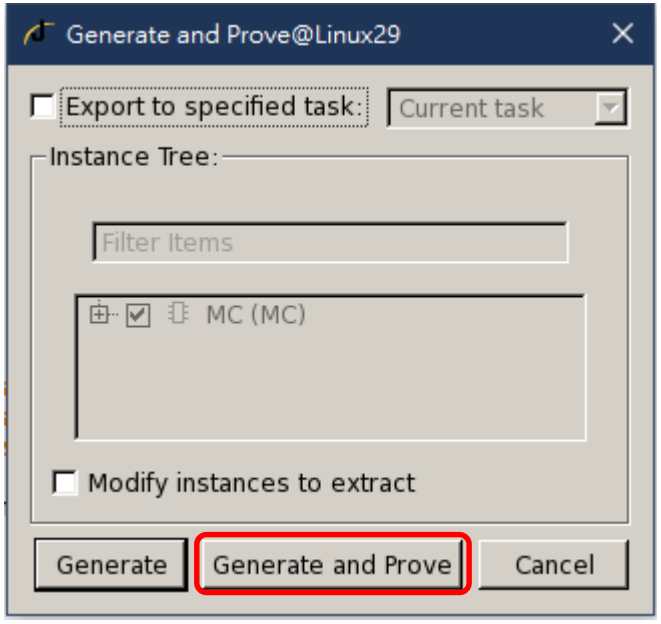
Setup the clock and Reset (cont.)

- **Setup the Reset**
  - press the button



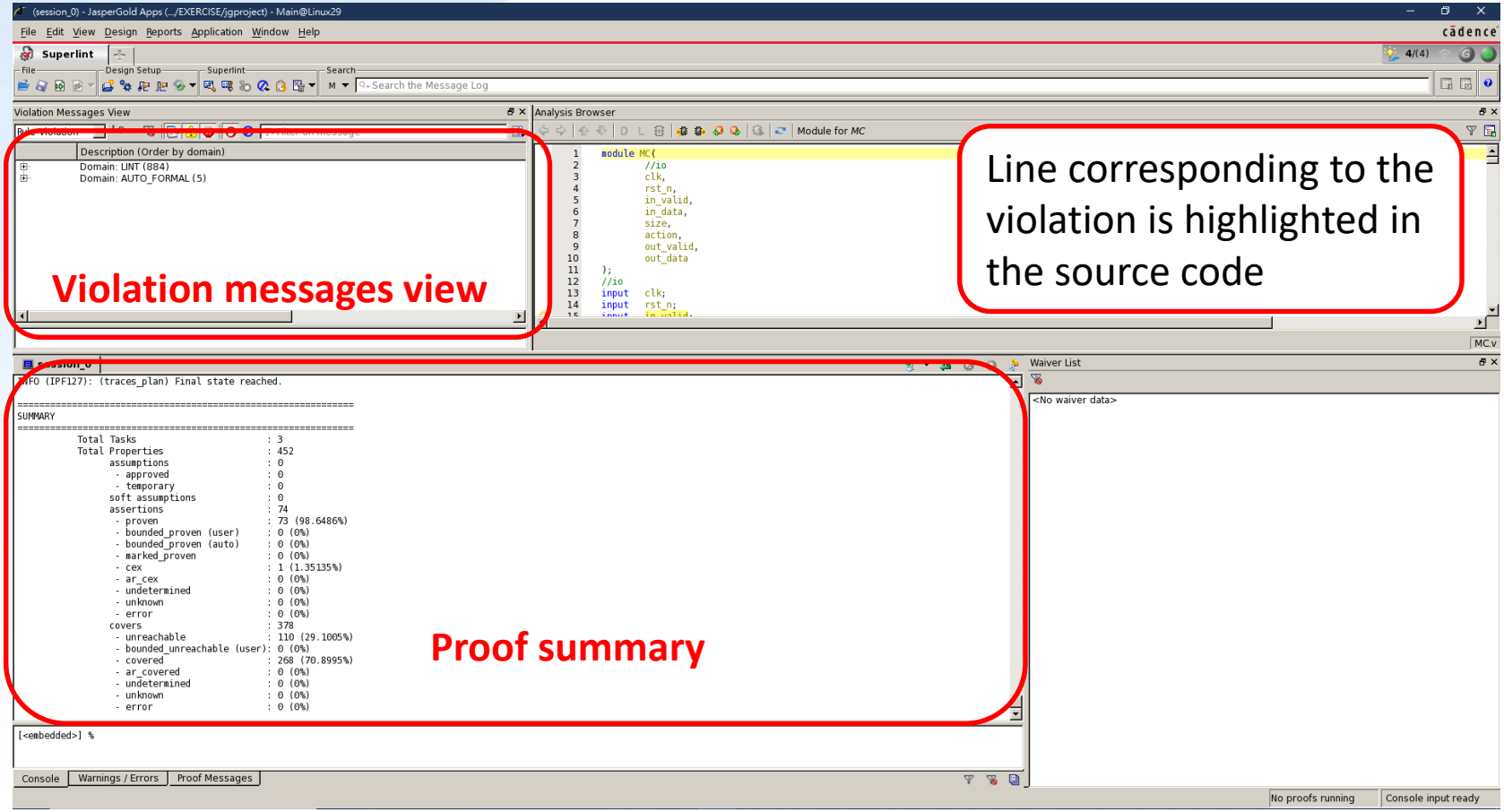
Extract and Prove Superlint Checks

- ✓ **Press the Button**



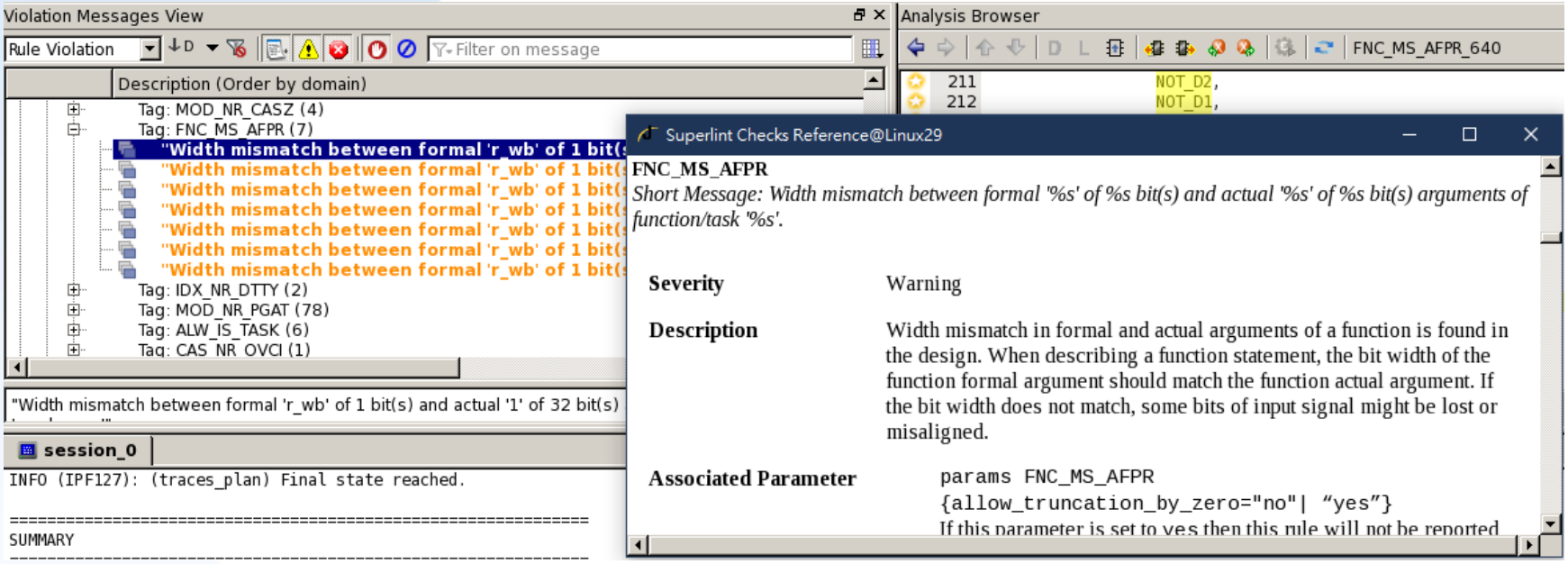
Extract and Prove Superlint Checks

- ✓ **Then you can see the violation messages**



Extract and Prove Superlint Checks

- ✓ **Check the violation description**



Double click the violation message to view the information