Homework Ⅲ 說明

Instructor：Lih-Yih Chiou

TA : Yun-Ru Chen

Date : 2020/10/28

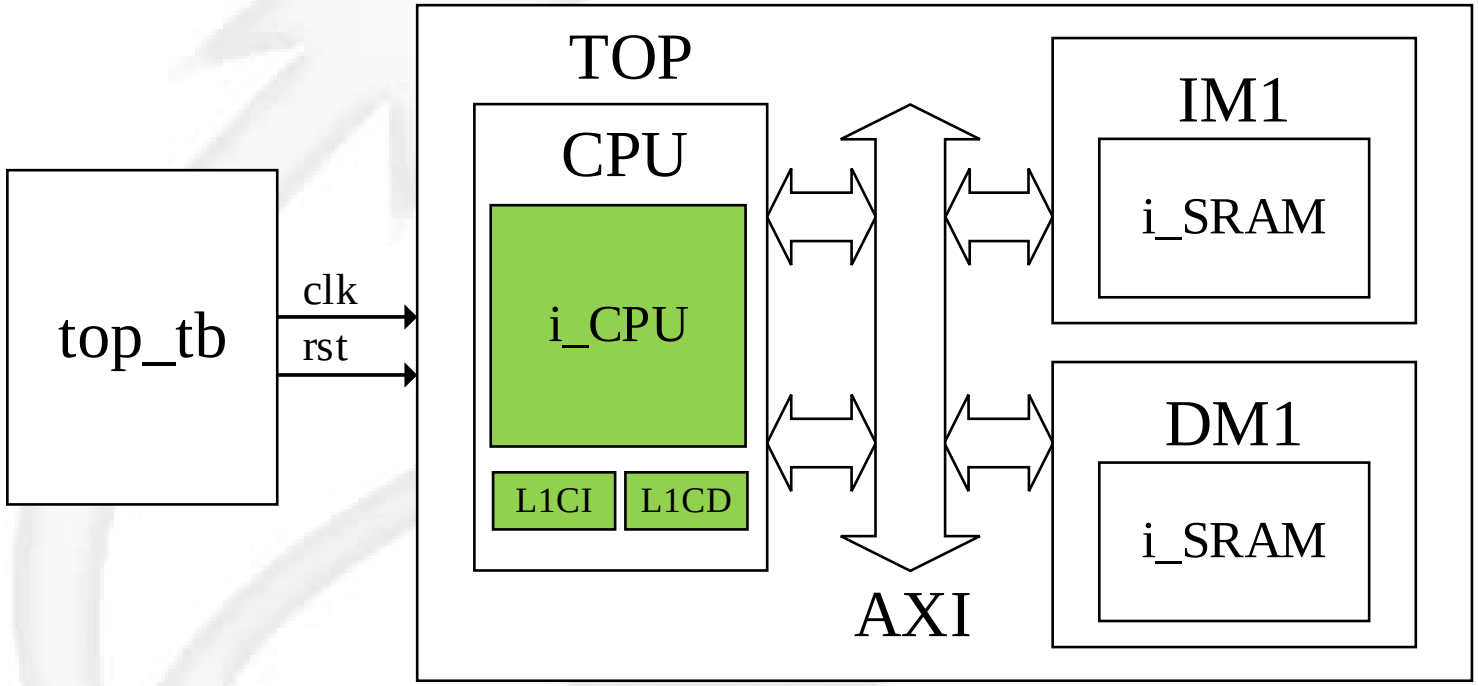
Outline

- System Architecture
- Problem 1
  - Specification
  - Verification
- Problem 2
  - Specification
  - Verification
- Submission rule

VLSI Design LAB

System Architecture

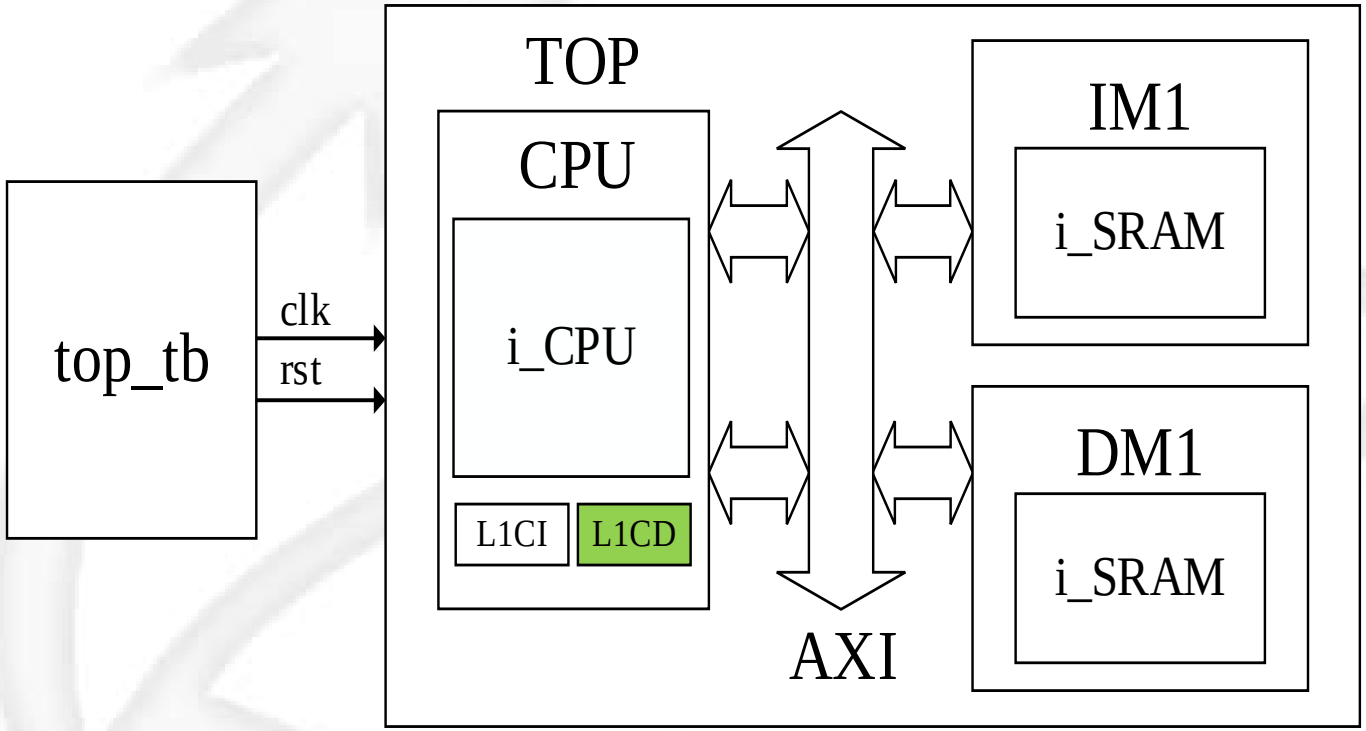
System Architecture



Problem 1

Specification

Specification(1/6)

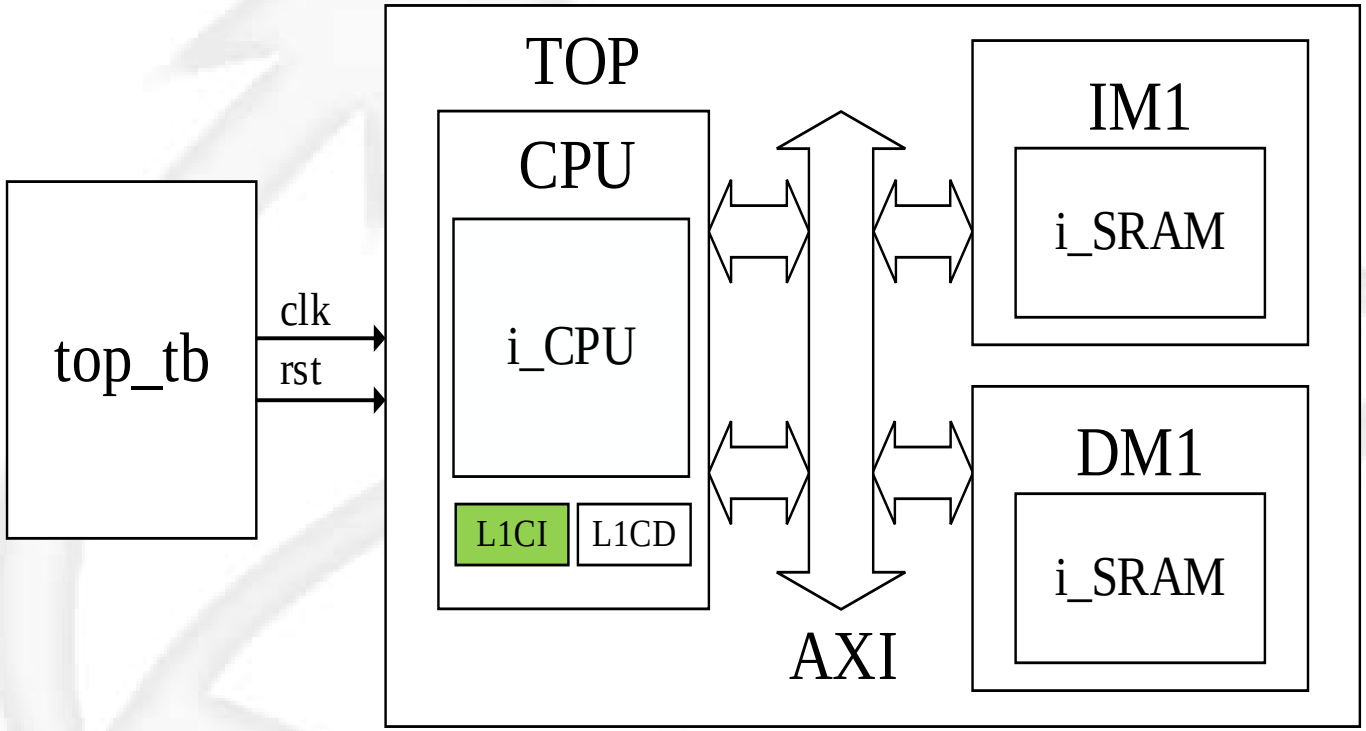


Specification(2/6)

L1C_data	clk	input	1	clock
	rst	input	1	reset (active high)
	core_addr	input	32	address from CPU
	core_req	input	1	memory access request from CPU
	core_write	input	1	write signal from CPU
	core_in	input	32	data from CPU
	core_type	input	3	write/read byte, half word, or word (listed in include/def.svh) from CPU
	D_out	input	32	data from CPU wrapper
	D_wait	input	1	wait signal from CPU wrapper
	core_out	output	32	data to CPU
	core_wait	output	1	wait signal to CPU
	D_req	output	1	request to CPU wrapper
	D_addr	output	32	address to CPU wrapper
	D_write	output	1	write signal to CPU wrapper
	D_in	output	32	write data to CPU wrapper
	D_type	output	3	write/read byte, half word, or word (listed in include/def.svh) to CPU wrapper
	valid	logic	64	valid bits of each cache line
	index	logic	6	address to tag array/data array
	TA_write	logic	1	write signal to tag_array
	TA_read	logic	1	read signal to tag_array
	TA_in	logic	22	write data to tag_array
	TA_out	logic	22	read data from tag_array
	DA_write	logic	16	write signal to data_array
	DA_read	logic	1	read signal to data_array
	DA_in	logic	128	write data to data_array
	DA_out	logic	128	read data from data_array

Inputs and outputs of module L1C_data/L1C_inst has declared in src/L1C_data.sv and src/L1C_inst.sv.
You can only add logics inside the modules.

Specification(3/6)



Specification(4/6)

L1C_inst	clk	input	1	clock
	rst	input	1	reset (active high)
	core_addr	input	32	address from CPU
	core_req	input	1	memory access request from CPU
	core_write	input	1	write signal from CPU
	core_in	input	32	data from CPU
	core_type	input	3	write/read byte, half word, or word (listed in include/def.svh) from CPU
	I_out	input	32	data from CPU wrapper
	I_wait	input	1	wait signal from CPU wrapper
	core_out	output	32	data to CPU
	core_wait	output	1	wait signal to CPU
	I_req	output	1	request to CPU wrapper
	I_addr	output	32	address to CPU wrapper
	I_write	output	1	write signal to CPU wrapper
	I_in	output	32	write data to CPU wrapper
	I_type	output	3	write/read byte, half word, or word (listed in include/def.svh) to CPU wrapper
	valid	logic	64	valid bits of each cache line
	index	logic	6	address to tag array/data array
	TA_write	logic	1	write signal to tag_array
	TA_read	logic	1	read signal to tag_array
	TA_in	logic	22	write data to tag_array
	TA_out	logic	22	read data from tag_array
	DA_write	logic	16	write signal to data_array
	DA_read	logic	1	read signal to data_array
	DA_in	logic	128	write data to data_array
	DA_out	logic	128	read data from data_array

Inputs and outputs of module L1C_data/L1C_inst has declared in src/L1C_data.sv and src/L1C_inst.sv.
You can only add logics inside the modules.

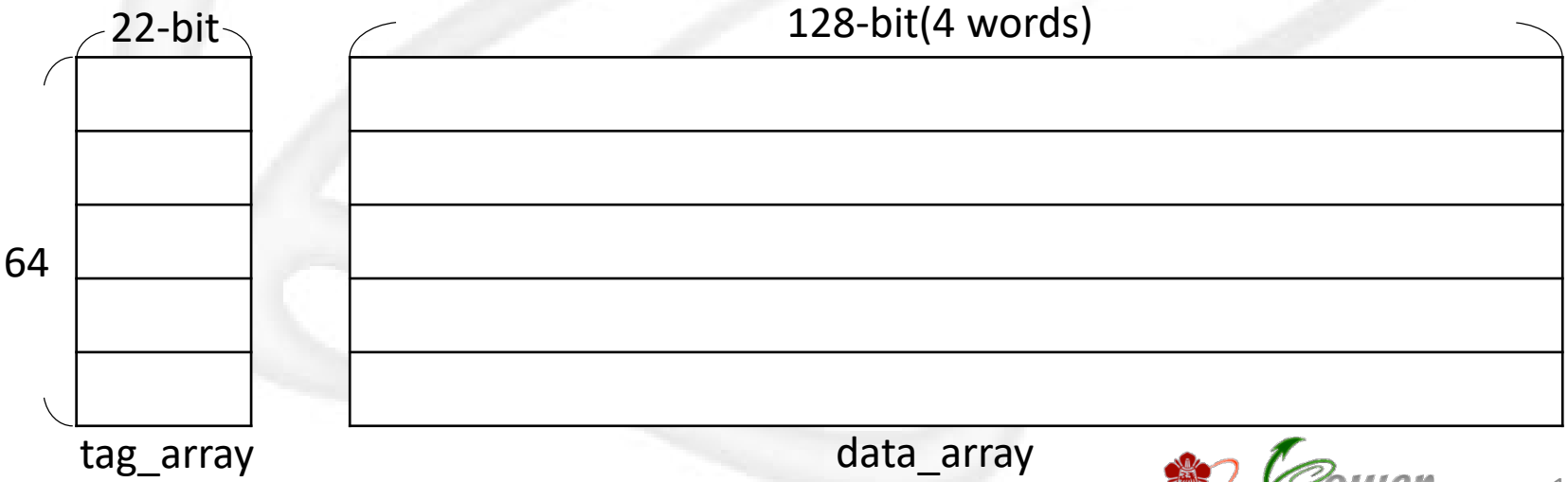
Specification(5/6)

Table 1-1: Cache actions on different condition

condition	core read	core write
hit	transmit data into core	write data into cache and memory
miss	read a line from memory	only write data into memory

Table 1-2: Array characteristics

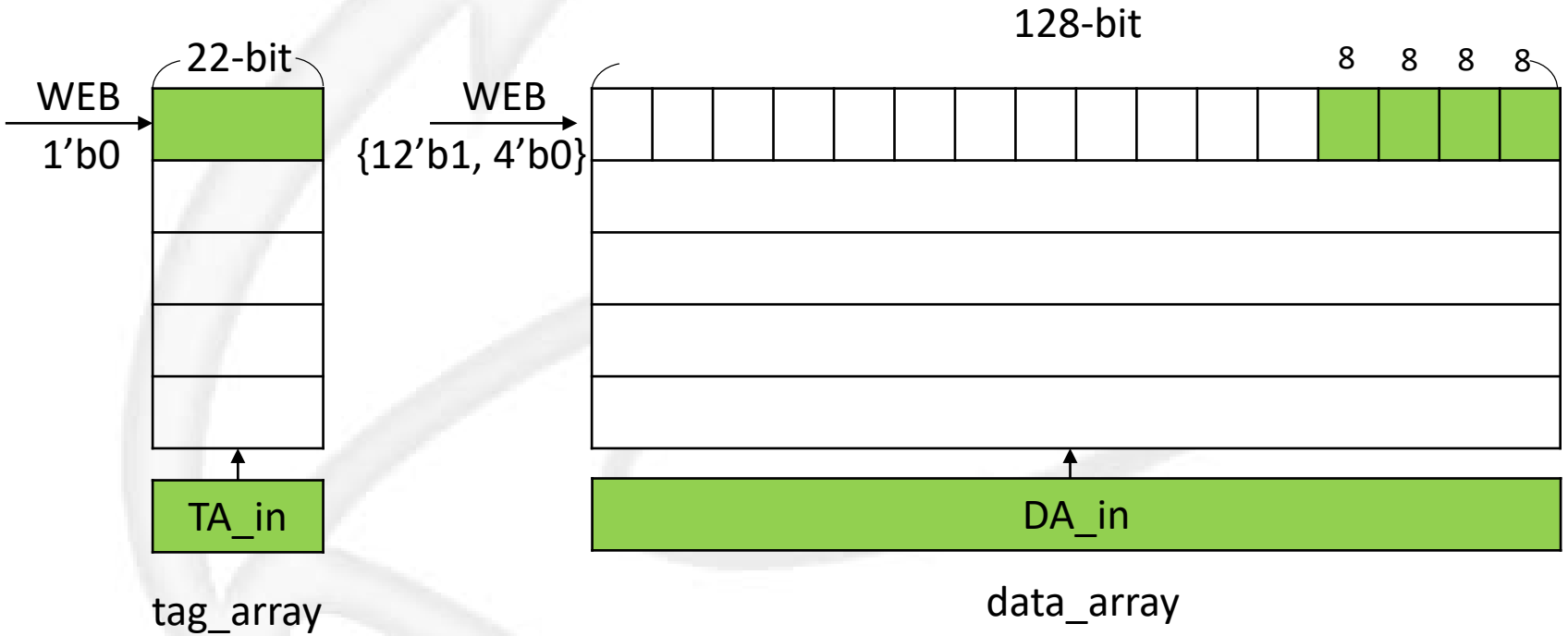
Type	Lines	Words per line	Bytes per word	Bits per line	Writing mode
data_array	64	4	4	128	Byte Write
tag_array		1		22	Word Write



Hit = valid && (TA_out equal core_addr[31:10])

Specification(6/6)

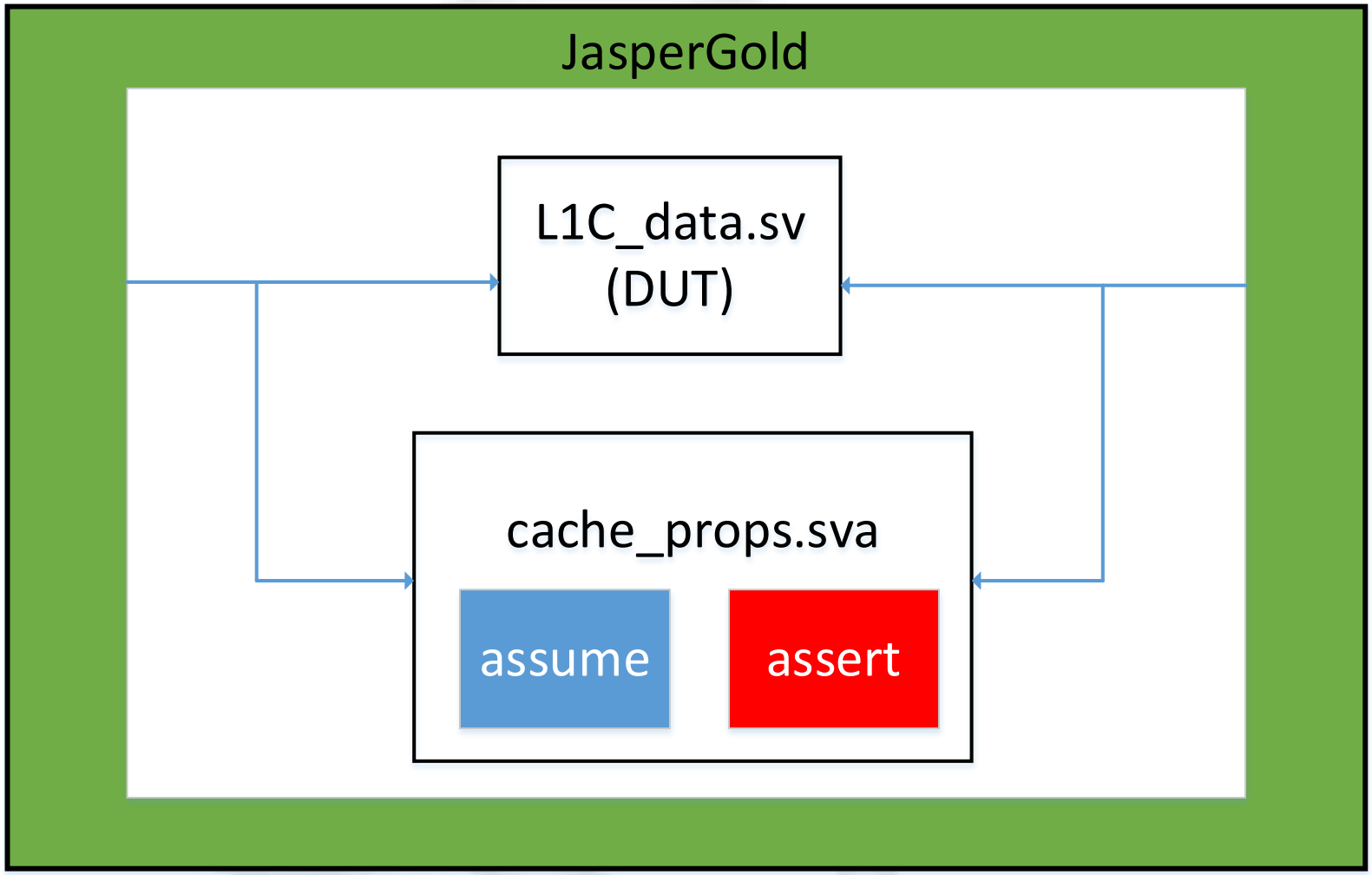
- Tag_array is word write, while data_array is byte write



Problem 1

Verification

Verification(1/3)



Verification(3/3)

- Don’t modify cache_props.sva
- Use JasperGold to verify L1 data cache(L1C_data.sv) by command “make jg”
- The should be no assertion violations in the result

Problem 2

Specification

Specification(1/3)

- Implement 4 new instructions on your CPU
  - S-type

31 25	24 20	19 15	14 12	11 7	6 0		
imm[11:5]	**rs2**	**rs1**	**funct3**	**imm[4:0]**	**opcode**	**Mnemonic**	**Description**
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH	$M[rs1+imm]_h = rs2_h$

  - I-type

31 20	19 15	14 12	11 7	6 0		
imm[11:0]	**rs1**	**funct3**	**rd**	**opcode**	**Mnemonic**	**Description**
imm[11:0]	rs1	001	rd	0000011	LH	$rd = M[rs1+imm]_{hs}$
imm[11:0]	rs1	100	rd	0000011	LBU	$rd = M[rs1+imm]_{bu}$
imm[11:0]	rs1	101	rd	0000011	LHU	$rd = M[rs1+imm]_{hu}$

1. Byte = 8 bits
2. half-word = 16 bits
3. SH means store lowest half-word of rs2 to the memory
4. Data pre-processing and post-processing should be done in CPU

Specification(2/3)

- Module name須符合下表要求

Category	Name			
	File	Module	Instance	SDF
RTL	top.sv	top	TOP	
Gate-Level	top_syn.v	top	TOP	top_syn.sdf
Physical	top_pr.sv	top	TOP	top_pr.sdf
RTL	L1C_inst.sv	L1C_inst	L1CI	
RTL	L1C_data.sv	L1C_data	L1CD	
RTL	SRAM_wrapper.sv	SRAM_wrapper	IM1	
RTL	SRAM_wrapper.sv	SRAM_wrapper	DM1	
RTL	SRAM_rtl.sv	SRAM	i_SRAM	
RTL	tag_array_wrapper.sv	tag_array_wrapper	TA	
RTL	tag_array_rtl.sv	tag_array	i_tag_array	
RTL	data_array_wrapper.sv	data_array_wrapper	DA	
RTL	data_array_rtl.sv	data_array	i_data_array	

- 助教提供好的input/output ports請勿更動
- 紫色部分為助教已提供或已定義好，請勿任意更改
- 其餘部分需按照要求命名，以免testbench抓不到正確的名稱

Specification(3/3)

- Tag_array.v and data_array.v already include in top_tb.sv, don't re-include in your design
- Synthesize and do APR of HW3
  - Synthesize command: make synthesize
  - APR command: make innovus

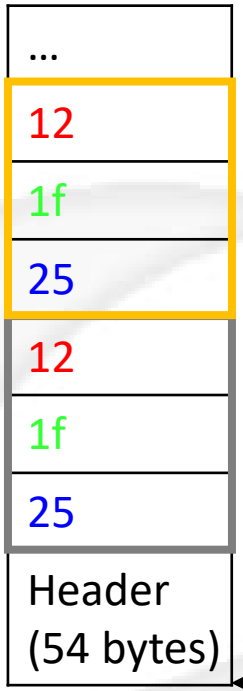
Problem 2

Verification

Verification(1/2)

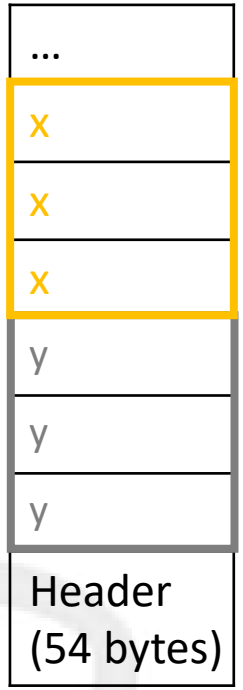
- prog0
  - ➔ 測試37個instruction (助教提供)
- prog1
  - ➔ Sort algorithm of half-word
- Prog2
  - ➔ Gray scale

Verification(2/2)



x = 0.11*b + 0.59*g + 0.3*r
= 0.11*25 + 0.59*1f + 0.3* 12

y = 0.11*b + 0.59*g + 0.3*r
= 0.11*25 + 0.59*1f + 0.3* 12



B M Size of BMP file (byte)

The number of bits per pixel

Address	0	1	2	3	4	5	6	7	8	9	a	b	Dump
00000000	42	4d	36	00	24	00	00	00	00	00	36	00	BM6.$.....6.
0000000c	00	00	28	00	00	00	00	04	00	00	00	03	..(.....
00000018	00	00	01	00	18	00	00	00	00	00	00	00	.....
00000024	24	00	c4	0e	00	00	c4	0e	00	00	00	00	$.?..?....
00000030	00	00	00	00	00	00	25	1f	12	25	1f	12	......%..%..
0000003c	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
00000048	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
00000054	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
00000060	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
0000006c	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..
00000078	25	1f	12	25	1f	12	25	1f	12	25	1f	12	%..%..%..%..

從檔案頭到點陣圖資料須偏移的byte數

BMP檔案解析參考資料:
https://www.itread01.com/content/1549504280.html

Submission rule

Report

- 請使用附在檔案內的Submission Cover
- 請勿將code貼在.docx內 (program的程式可截圖說明)
  - 請將.sv包在壓縮檔內，不可截圖於.docx中
- 需要Summary及Lessons learned(Summary table請放在第二頁，清楚列出有完成以及沒完成的部分)
- 若兩人為一組，須寫出貢獻度(貢獻度請放第二頁)
  - Ex: A(N26071234) 55%, B(N26075678) 45%
  - Total 100%
  - 自己一組則不用寫

檔案結構

- 參考 Appendix A
- sim/CYCLE
  - Specify your clock cycle time
- sim/MAX
  - Specify max clock cycle number
- sim/prog0
  - Don’t modify contents
- sim/progX (X ≠ 0)
  - main.S
  - main.c
    - Submit one of these

繳交期限

- 2020/12/2 (三) 14:00前上傳
  - 不接受遲交，請務必注意時間
  - Moodle只會留存你最後一次上傳的檔案，檔名只要是「*N26XXXXXX.tar*」即可，不需要加上版本號

VLSI Design LAB

Thanks for your participation and attendance !!