

HOMEWORK IV

Due day: 23:59 1. 1 (Wednesday), 2025

Introduction

In this assignment, your responsibility includes completing the CPU design, incorporating **instruction and data cache** elements, addressing **AXI clock domain crossing** challenges with various memory or IP components, and executing automatic place and route (**APR**).

Finally you need to write a simple program to booting your CPU using DMA. The CPU and the memories (ROM, SRAM and DRAM), WDT.

General rules for deliverables

- This homework needs to be completed by INDIVIDUAL student or a TEAM. Only one submission is needed for a team. You **MUST** write down you and your teammate's name on the submission cover of the report. Otherwise duplication of other people's work may be considered cheating.
- Compress all files described in the problem statements into one **tar** file.
- Submit the compressed file to the course website before the due day.

Warning! **AVOID** submitting in the last minute. Late submission is not accepted.

Grading Notes

- **Important!** **DO** remember to include your SystemVerilog code. **NO** code, **NO** grades. Also, if your code can not be recompiled by TA successfully using tools in SoC Lab and commands in Appendix B, you will receive **NO** credit.
- Write your report seriously and professionally. Incomplete description and information will reduce your chances to get more credits.
- If extra works (like synthesis, post-simulation or additional instructions) are done, please describe them in your final report clearly for bonus points.
- Please follow course policy.
- Verilog and System Verilog generators aren't allowed in this course.

Deliverables

1. All SystemVerilog codes including components, testbenches and machine codes for

HOMEWORK IV

each lab exercise. NOTE: Please **DO NOT** include source codes in the report!

2. Write a homework report in MS word and follow the convention for the file name of your report: N260xxxxx.docx. Please save as docx file format and replace N260xxxxx with your student ID number. (Let the letter be uppercase.) **If you are a team, you should name your report, top folder and compressed file with the student ID number of the person uploading the file. The other should be written on the submission cover of your report, or you will receive NO credit.**
3. Organize your files as the hierarchy in Appendix A.

Report Writing Format

- a. Use the **submission cover** from the course website.
- b. **A summary in the beginning** to state what has been done.
- c. Report requirements from each problem.
- d. Describe the major **problems** you encountered and your resolutions.
- e. **APR flow** explanation
- f. **Lessons learned** from this homework.

HOMEWORK IV

Problems

1.1 Problem Description (AXI clock domain crossing)

In complex systems, various components may operate at different clock frequencies to optimize their performance and power consumption. AXI clock domain crossing enables seamless communication between these components, allowing them to exchange data without timing mismatches or data corruption. In this assignment, a significant focus will be on managing **AXI clock domain crossing** challenges. The task involves addressing intricacies associated with signals transitioning between different clock domains within the AXI interface to ensure proper synchronization and reliable communication between components.

1.2 Problem Description (cache)

In this assignment, you are tasked with enhancing the performance of a basic CPU architecture by integrating a cache memory subsystem. The goal is to implement a cache hierarchy that includes **both instruction and data caches beneath the CPU**. The specific details of the CPU and cache configurations are provided, and your objective is to seamlessly integrate the cache into the existing CPU architecture.

1.3 Block Overview

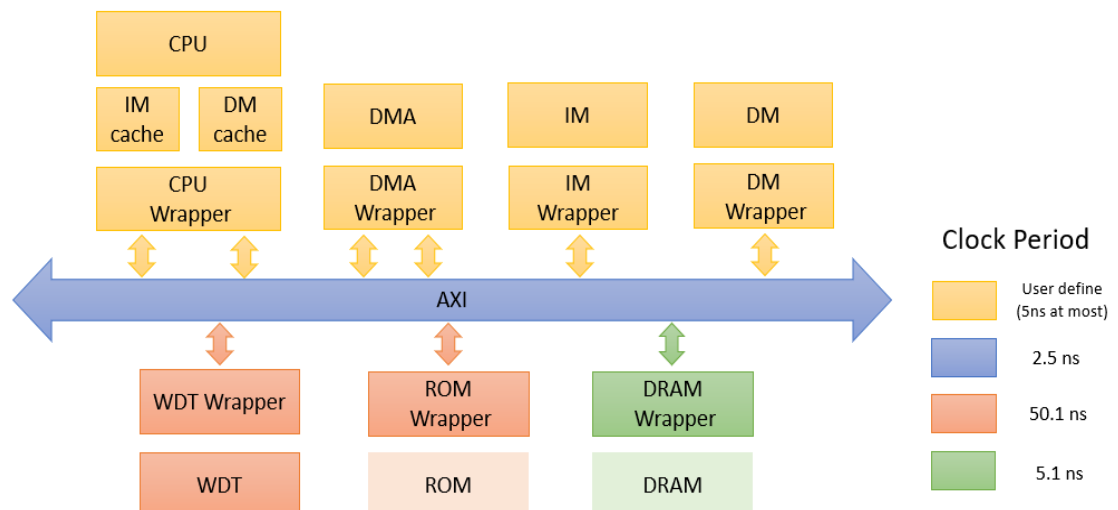


Fig. 1-1: System block diagram

HOMEWORK IV

1.4 Module Specification

Table 1-1: Module naming rule

Category	Name			
	File	Module	Instance	SDF
RTL	CHIP.v	CHIP	chip	
Gate-Level	CHIP_syn.v		chip	chip_syn.sdf
Physical	CHIP_pr.v		chip	chip_pr.sdf
RTL	top.sv	top	u_TOP	
RTL	L1C_inst.sv	L1C_inst	L1CI	
RTL	L1C_data.sv	L1C_data	L1CD	
RTL	AXI.sv	AXI	AXI	
RTL	SRAM_wrapper.sv	SRAM_wrappe r	IM1	
RTL	SRAM_wrapper.sv	SRAM_wrappe r	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	
Behavior	ROM.v	ROM	i_ROM	
Behavior	DRAM.v	DRAM	i_DRAM	

Table 1-2: Module signals

Module	Specifications			
	Name	Signal	Bits	Function explanation
top, CHIP (with iopad)	System signals			
	cpu_clk	input	1	Clock for cpu
	axi_clk	input	1	AXI clock
	rom_clk	input	1	ROM clock ,WDT
	dram_clk	input	1	DRAM clock

HOMEWORK IV

	cpu_rst	input	1	Reset active high
	axi_rst	input	1	Reset active high
	rom_rst	input	1	Reset active high
	dram_rst	input	1	Reset active high
	Connect with ROM			
	ROM_out	input	32	Data from ROM
	ROM_read	output	1	ROM output enable
	ROM_enable	output	1	Enable ROM
	ROM_address	output	12	Address to ROM
	Connect with DRAM			
	DRAM_Q	input	32	Data from DRAM
	DRAM_valid	input	1	DRAM output data valid
	DRAM_CS _n	output	1	DRAM Chip Select (active low)
	DRAM_WE _n	output	4	DRAM Write Enable (active low)
	DRAM_RAS _n	output	1	DRAM Row Access Strobe (active low)
	DRAM_CAS _n	output	1	DRAM Column Access Strobe (active low)
ROM	DRAM_A	output	11	Address to DRAM
	DRAM_D	output	32	Data to DRAM
	System signals			
	CK	input	1	System clock
	Memory ports			
	DO	output	32	ROM data output
	OE	input	1	Output enable (active high)

HOMWORK IV

	CS	input	1	Chip select (active high)
	A	input	12	ROM address input
	Memory Space			
	Memory_byte0	reg	8	Size: [0:4095]
	Memory_byte1	reg	8	Size: [0:4095]
	Memory_byte2	reg	8	Size: [0:4095]
	Memory_byte3	reg	8	Size: [0:4095]
DRAM	System signals			
	CK	input	1	System clock
	RST	input	1	System reset (active high)
	Memory ports			
	CSn	input	1	DRAM Chip Select (active low)
	WEn	input	4	DRAM Write Enable (active low)
	RASn	input	1	DRAM Row Access Strobe (active low)
	CASn	input	1	DRAM Column Access Strobe (active low)
	A	input	11	DRAM Address input

DMA	System signals			
	clk	input	1	System clock
	rst	input	1	System reset (active high)
	DMAEN	input	1	Enable the DMA
	DMA_SRC	input	32	Source address of DMA
	DMA_DST	input	32	Destination address of DMA
	DMA_LEN	input	32	Total length of the data
	DMA_interrupt	output	1	DMA interrupt

HOMEWORK IV

Table 1-3: Slave configuration

Name	Number	Start address	End address
ROM	Slave 0	0x0000_0000	0x0000_1FFF
IM	Slave 1	0x0001_0000	0x0001_FFFF
DM	Slave 2	0x0002_0000	0x0002_FFFF
DMA	Slave 3	0x1002_0000	0x1002_04FF
WDT	Slave 4	0x1001_0000	0x1001_03FF
DRAM	Slave 5	0x2000_0000	0x201F_FFFF

Table 1-4: Clock domain

Name	Cycle period (ns)	clock domain
dram_clk	5.1	Dram
rom_clk	50.1	Rom,WDT
axi_clk	2.5	AXI
cpu_clk	User define (less than 5ns)	CPU, IM cache, DM cache,

You **SHOULD** use the timing constraint file, *DC.sdc*, provided in the course website to synthesize your top.sv. **Don't modify any constraint except clock period**. Your physical design should has following features:

- Use *Default.globals* as your global variable file. It will use *MMMC.view* as your analysis configuration and use *APR.sdc* as your timing constraint file.
- Don't modify the timing constraint in *APR.sdc* except clock period.**
Maximum clock period is 5 ns.
- Do Macro layout only. Don't add IO pad and bonding pad.
- The width of power ring is fixed to **3 μ m**. Add **three wire group**.
- The width of power stripe is fixed to **2 μ m**. At least add **one group for each direction**.

HOMEWORK IV

- f. The width of block ring is fixed to **3 μ m**.
- g. Don't add dummy metal.
- h. Must **add core filler**.
- i. Pass DRC and LVS check without any violation.

Your RTL code needs to comply with **Superlint within 95%** of your code, i.e., the number of errors & warnings in total shall not exceed 5% of the number of lines in your code. HINT: You can use the command in Appendix B to get the number of lines in your code. Remember to exclude *top_tb.sv*.

1.5 Verification

You should complete following programs and use the commands in Appendix B to verify your design.

- a. For **prog0**, **prog1**, **prog2**, **prog3**, you should write a boot program defined as *boot.c* to copy data between *_dram_i_start* and *_dram_i_end* to *_imem_start*, from *__data_paddr_start* to *_data_start* and *_data_end*, also from *sdata_paddr_start* to *sdata_start* and *sdata_end*. The booting program should be stored at ROM. Explain the *boot.c*.
- b. For **prog0**, use *main.S* to perform verification for the functionality of instructions. Show the terminal result and waveform in the report. The waveform should include new added instructions, and please explain the operation.
- c. Use **prog1** to perform floating point computation.
- d. Write a program defined as **prog2** to perform the matrix multiplication. The row size & column size of matrix is stored at the address named *array_size_i*, *array_size_j* and *array_size_k* in ".rodata" section defined in *data.S*. The first element is stored at the address named *array_addr* in ".rodata" section defined in *data.S*, others are stored at adjacent addresses. All elements in matrix are **signed 2-byte half-word** and you should store result byte by byte from "*_test_start*" to "*test_start*" + *array_size_i***array_size_j*-1.

HOMEWORK IV

- e. For *prog3*, when WDT is enabled, WDT counter starts to count. First time CPU executes to self-loop instruction until WDT times out. Then WDT will interrupt CPU to restart by ISR procedure. Second time CPU regularly restarts the watchdog timer to prevent it from timing out. **You shouldn't modify the interrupt service routine and the main program.**
- f. For CDC check, Use Spyglass CDC to verify the correctness of your design. You should show the results of Spyglass CDC. If there are warning reports, please explain clearly in your report. **Make sure that no error or fatal message in your design.**

1.6 Report Requirements

Your report should have the following features:

- a. **USE the provided .DOCX document from the TA. (otherwise get 0 credit of this homework)**
- b. **Do not change top_tb.sv, top_tb_WDT.sv and Makefile**
- c. Proper explanation of your design is required for full credits.
- d. Block diagrams shall be drawn to depict your designs.
- e. Show your screenshots of **the waveforms and the simulation results on the terminal(RTL,SYN,APR)** for the different test cases in your report and **illustrate** the correctness of your results.
- f. Show your screenshots of the Spyglass CDC reports and explain why your CDC circuit can work correctly.
- g. Show your IM/ DM cache **hit rate**, and explain
- h. Show your snapshots of **Floorplan View**, **Amoeba View** and **Physical View** in Innovus. Also, show the results of **Geometry Verification**, **Connectivity Verification**, and **Antenna Verification** have no violation.
 - If there are some violations, please explain the meaning of the violation
- i. Report the number of lines of your RTL code, the final results of running Superlint and 3~5 most frequent warning/errors in your code. Describe how you modify your code to comply with Superlint.
- j. Report and show screenshots of your prog0 to prog3 simulation time after synthesis and total cell area of your design. **20%** homework credit will be given based on your design performance & area.
- k. APR flow explanation in your report.
- l. Lesson learned

HOMEWORK IV

Appendix

A. File Hierarchy Requirements

All homework **SHOULD** be uploaded and follow the file hierarchy and the naming rules, especially the uppercase and the lowercase, specified below. You should create a main folder named your student ID number. It contains your homework report and every subfolder of the problems. The names of the files and the folders are labeled in red color, and the specifications are labeled in black color.

ATTENTION !!!!!!!

The file marked with a yellow highlighter is a non-editable file.

Fig. A-1 File hierarchy






- *N260XXXXX.tar* (**Don't** add version text in filename, e.g. *N260XXXXX_v1.tar*)
 - 📁 *N260XXXXX* (Main folder of this homework)
 - 📄 *N260XXXXX.docx* (Your homework report)
 - 📄 *StudentID* (Specify your student ID number in this file)
 - 📄 *StudentID2* (Specify your partner's student ID number in this file.
Please delete it if you don't have partner)
 - 📄 *Makefile* (You shouldn't modify it)
 - 📁 *src* (Your RTL code with *sv* format)
 - 📄 *CHIP.v*
 - 📄 *top.sv*
 - 📄 *LIC_inst.sv*
 - 📄 *LIC_data.sv*
 - 📄 *SRAM_wrapper.sv*
 - 📄 *tag_array_wrapper.sv*
 - 📄 *data_array_wrapper.sv*
 - 📄 *ROM_wrapper.sv*
 - 📄 *DRAM_wrapper.sv*
 - 📄 *sctrl_wrapper.sv*
 - 📄 *DMA.sv*
 - 📄 Other submodules (*.sv)
 - 📁 *AXI*
 - 📄 *AXI.sv*
 - 📄 Submodules of AXI (*.sv)

HOMEWORK IV








- 📁 *include* (Your RTL definition with *svh* format)
 - 📄 *AXI_def.svh*
 - 📄 *def.svh*
 - 📄 Definition files (*.svh)
- 📁 *syn* (Your synthesized code and timing file)
 - 📄 *top_syn.v*
 - 📄 *top_syn.sdf*
- 📁 *pr* (Your post-layout netlist and timing file)
 - 📄 *top_pr.v*
 - 📄 *top_pr.sdf*
 - 📄 *top_pr.gds*
- 📁 *script* (Any scripts of verification, synthesis or place and route)
 - 📄 script files (*.sdc, *.tcl or *.setup)
- 📁 *sim* (Testbenches and memory libraries)
 - 📄 *top_tb.sv* (Main testbench. You shouldn't modify it)
 - 📄 *top_tb_WDT.sv* (WDT testbench. You shouldn't modify it)
 - 📄 *CYCLE_MAX* (Specify your clock cycle time and max clock cycle number in this file)
- 📁 *SRAM* (SRAM libraries and behavior models)
 - 📄 Library files (*.lib, *.db, *.lef or *.gds)
 - 📄 *SRAM.ds* (SRAM datasheet)
 - 📄 *SRAM_rtl.sv* (SRAM RTL model)
 - 📄 *SRAM.v* (SRAM behavior model)
- 📁 *ROM* (ROM behavior models)
 - 📄 *ROM.v* (ROM behavior model)
- 📁 *DRAM* (DRAM behavior models)
 - 📄 *DRAM.v* (DRAM behavior model)
- 📁 *data_array* (data_array libraries and behavior models)
 - 📄 Library files (*.lib, *.db, *.lef or *.gds)
 - 📄 *data_array.ds* (data_array datasheet)
 - 📄 *data_array_rtl.sv* (data_array RTL model)
 - 📄 *data_array.v* (data_array behavior model)
- 📁 *tag_array* (tag_array libraries and behavior models)
 - 📄 Library files (*.lib, *.db, *.lef or *.gds)
 - 📄 *tag_array.ds* (tag_array datasheet)
 - 📄 *tag_array_rtl.sv* (tag_array RTL model)
 - 📄 *tag_array.v* (tag_array behavior model)

HOMEWORK IV









prog0 (Subfolder for Program 0)

-  *Makefile* (Compile and generate memory content)
-  *main.S* (Assembly code for verification)
-  *setup.S* (Assembly code for testing environment setup)
-  *link.ld* (Linker script for testing environment)
-  *golden.hex* (Golden hexadecimal data)









prog1 (Subfolder for Program 1)


-  *Makefile* (Compile and generate memory content)
-  *main.c* (C code for verification)
-  *boot.c* (C code for booting)
-  *isr.S* (Interrupt service routine)
-  *setup.S* (Assembly code for testing environment setup)
-  *link.ld* (Linker script for testing environment)
-  *golden.hex* (Golden hexadecimal data)

prog2 (Subfolder for Program 2)

-  *Makefile* (Compile and generate memory content)
-  *boot.c** (C code for verification)
-  *main.S** (Assembly code for verification)
-  *main.c** (C code for verification)
-  *data.S* (Assembly code for testing data)
-  *setup.S* (Assembly code for testing environment setup)
-  *link.ld* (Linker script for testing environment)
-  *golden.hex* (Golden hexadecimal data)

prog3 (Subfolder for Program 3)

-  *Makefile* (Compile and generate memory content)
-  *boot.c** (C code for verification)
-  *main.S** (Assembly code for verification)
-  *main.c** (C code for verification)
-  *isr.S* (Interrupt service routine)
-  *setup.S* (Assembly code for testing environment setup)
-  *link.ld* (Linker script for testing environment)
-  *golden.hex* (Golden hexadecimal data)

-  Any other files for your design, e.g. submodules and headers
- × **No waveform files allowed**, e.g. files of *fsdb* and *vcd* format
- × **No temporary files allowed**, e.g. *INCA_libs*, *ncverilog.log*, *novas**

B. Simulation Setting Requirements

HOMEWORK IV

You **SHOULD** make sure that your code can be simulated with specified commands in Table B-1. **TA will use the same command to check your design under SoC Lab environment. If your code can't be recompiled by TA successfully, you receive NO credit.** You can use macros in Table B-2 to help your verification.

HOMEWORK IV

Table B-1: Simulation commands

Simulation Level	Command
Problem1	
RTL	<code>make rtl_all</code>
Pre-layout Gate-level	<code>make syn_all</code>
Post-layout Gate-level	<code>make pr_all</code>

X stands for 0,1,2,3..., depend on which verification program is selected.

Table B-2: Makefile macros

Situation	Command
RTL simulation for progX	<code>make rtlX</code>
Post-synthesis simulation for progX	<code>make synX</code>
Post-layout simulation for progX	<code>make prX</code>
Dump waveform (no array)	<code>make {rtlX,synX, prX} FSDB=1</code>
Dump waveform (with array)	<code>make {rtlX,synX, prX} FSDB=2</code>
Open nWave without file pollution	<code>make nWave</code>
Open Superlint without file pollution	<code>make superlint</code>
Open DesignVision without file pollution	<code>make dv</code>
Synthesize your RTL code (You need write <i>synthesis.tcl</i> in <i>script</i> folder by yourself)	<code>make synthesize</code>
Open Innovus without file pollution	<code>make innovus</code>
Delete built files for simulation, synthesis or verification	<code>make clean</code>
Check correctness of your file structure	<code>make check</code>
Compress your homework to <i>tar</i> format	<code>make tar</code>
Run JasperGold VIP on AXI bridge without file pollution (RTL only)	<code>make vip_b</code>
Open Spyglass without file pollution	<code>make spyglass</code>

You can use the following command to get the number of lines:

```
wc -l src/* src/AXI/* include/*
```

C. RISC-V Instruction Format

Table C-1: Instruction type

☞ R-type

31	25	24	20	19	15	14	12	11	7	6	0
----	----	----	----	----	----	----	----	----	---	---	---

HOMEWORK IV

funct7	rs2	rs1	funct3	rd	opcode
--------	-----	-----	--------	----	--------

I-type

31	20	19 15	14 12	11	7	6	0
imm[31:20]		rs1	funct3	rd	opcode		

S-type

31	25	24	20	19	15	14	12	11	7	6	0
imm[11:5]		rs2		rs1		funct3		imm[4:0]		opcode	

B-type

31	30	25	24	20	19	15	14	12	11	8	7	6	0
imm[12]	imm[10:5]		rs2	rs1	funct3		imm[4:1]		imm[11]		opcode		

U-type

31	12	11	7	6	0
imm[31:12]		rd		opcode	

J-type

31	30	21	20	19	12	11	7	6	0
imm[20]	imm[10:1]		imm[11]	imm[19:12]		rd		opcode	

Table C-2: Immediate type

I-immediate

31	11	10	5	4	1	0
— inst[31] —		inst[30:25]		inst[24:21]		inst[20]

S-immediate

31	11	10	5	4	1	0
— inst[31] —		inst[30:25]		inst[11:8]		inst[7]

B-immediate

31	12	11	10	5	4	1	0
— inst[31]—		inst[7]	inst[30:25]	inst[11:8]		0	

U-immediate

31	30	20	19	12	11	0
Inst[31]	inst[30:20]		inst[19:12]		— 0 —	

J-immediate

31	20	19	12	11	10	5	4	1	0
— inst[31] —		inst[19:12]		inst[20]	inst[30:25]		inst[24:21]		0

“— X —” indicates that all the bits in this range is filled with X.