

Homework II 說明

Instructor : Lih-Yih Chiou

TA : Tzung Jin Tsai

Date : 10/09/2024



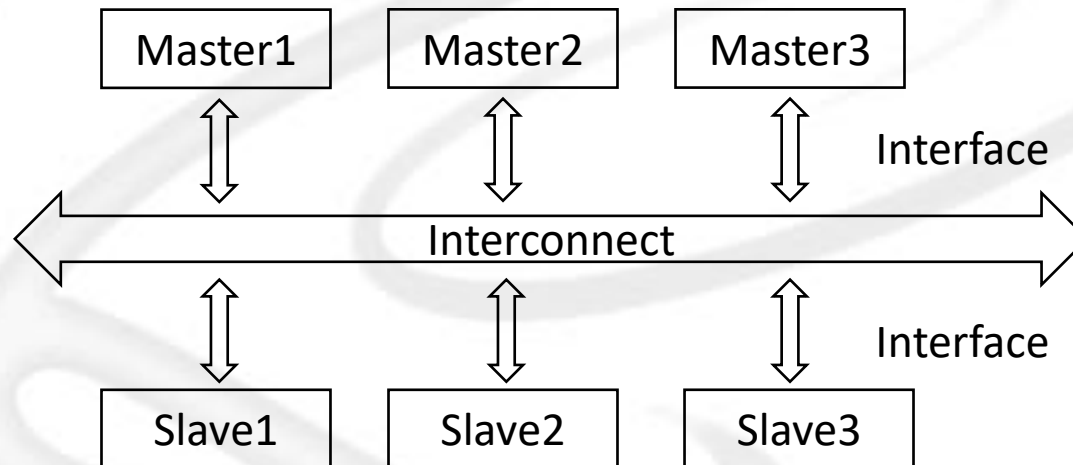
Outline

- 作業內容說明
- Problem 1
 - ➔ 作業驗證說明
- Problem 2
 - ➔ 作業驗證說明
- 作業繳交注意事項

作業内容説明

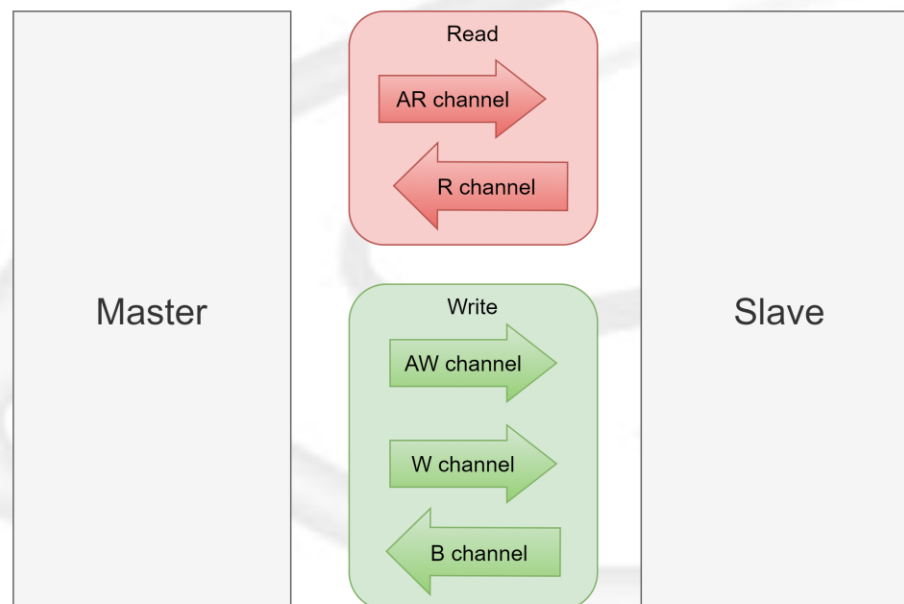
Bus Architecture

- Enable communication and data exchange between various hardware components, ensuring their co-operation
- Various protocols depends on the specific hardware design
 - ➔ APB
 - ➔ AHB
 - ➔ AXI (main focus)



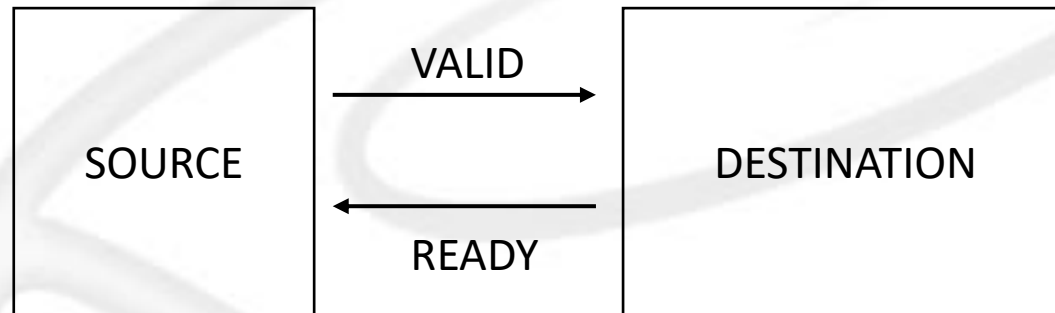
AXI4 introduction

- Advanced bus communication protocol developed by ARM, widely used in System-on-Chip (SoC) and FPGA-based system designs.
- Feature
 - ➔ Separated channels
 - ➔ Burst transfer
 - ➔ Support multiple outstanding transactions
 - ➔ Support out-of-order transactions



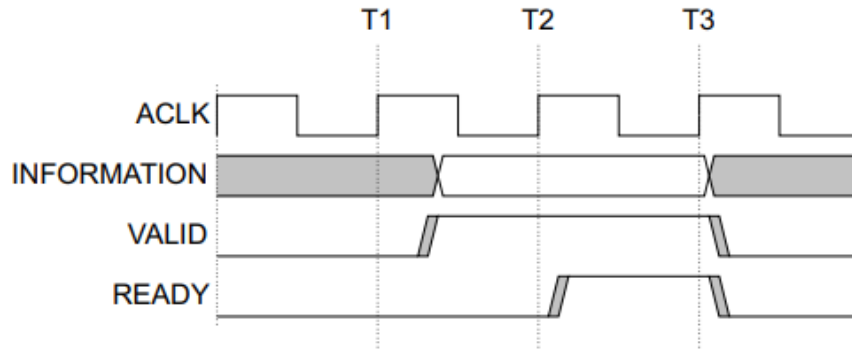
Handshake process

- ❑ Each channel uses **VALID/READY handshake** process to transfer address, data, and control information.
- ❑ Transfer occurs only when both the VALID and READY signals are HIGH

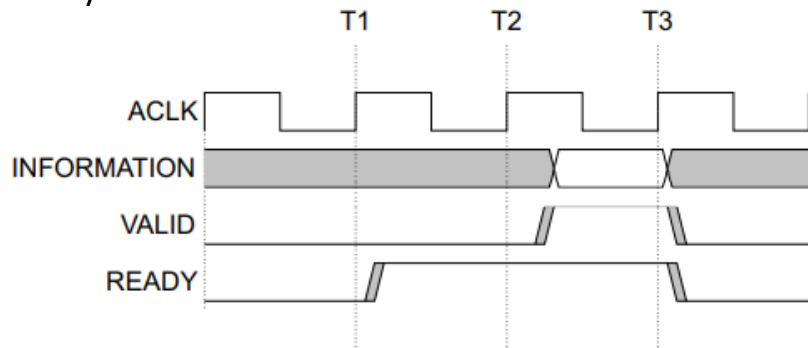


Handshake process

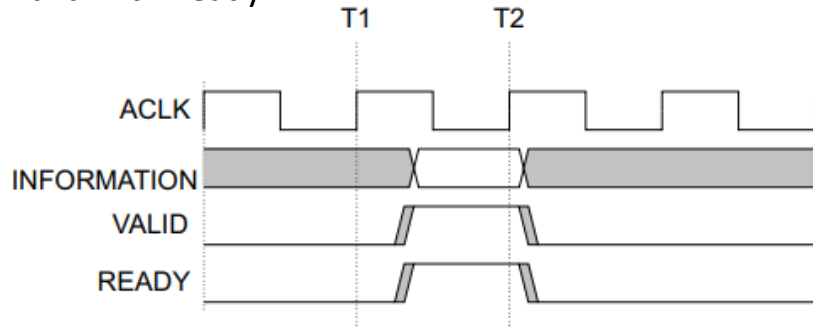
Case1 : Valid before Ready



Case2 : Ready before Valid



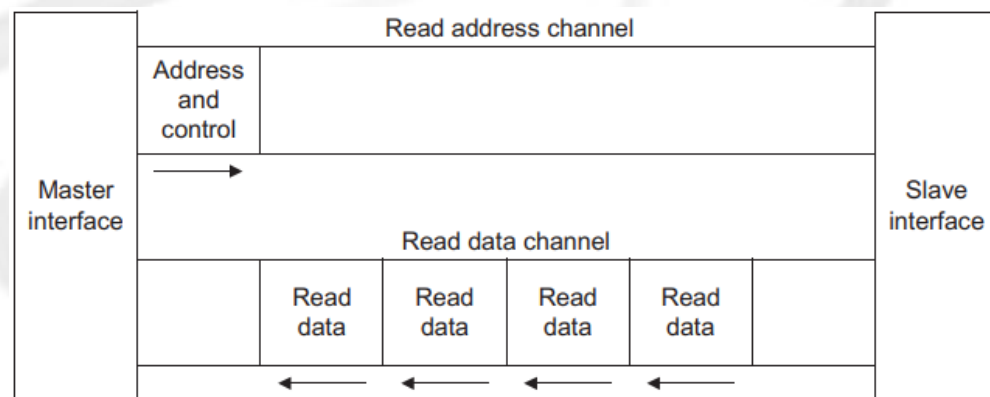
Case3 : Valid with Ready



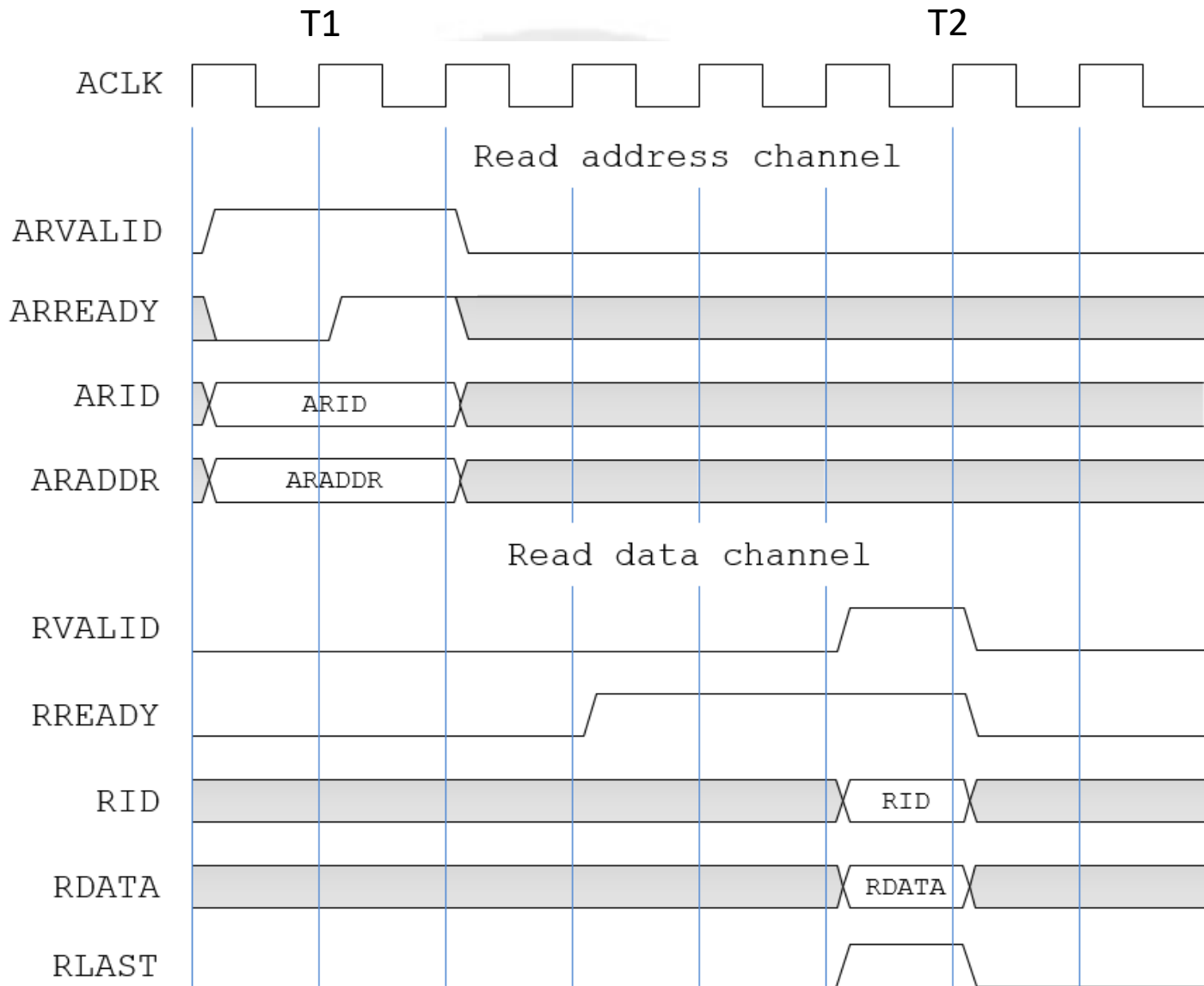
VALID signal must not be dependent on the READY signal in the transaction!!

Read Transaction

- ❑ Two channels for read process
 - ➔ Read Address Channel (AR)
 - ➔ Read Data Channel (R)
- ❑ Each channel has valid and ready signal for HS process

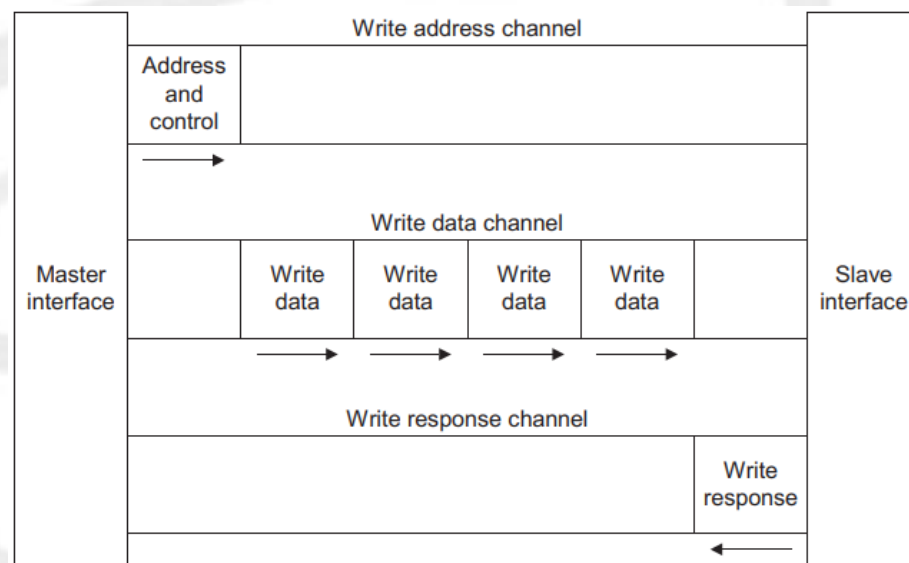


AXI Read Transfer (Burst length = 1)

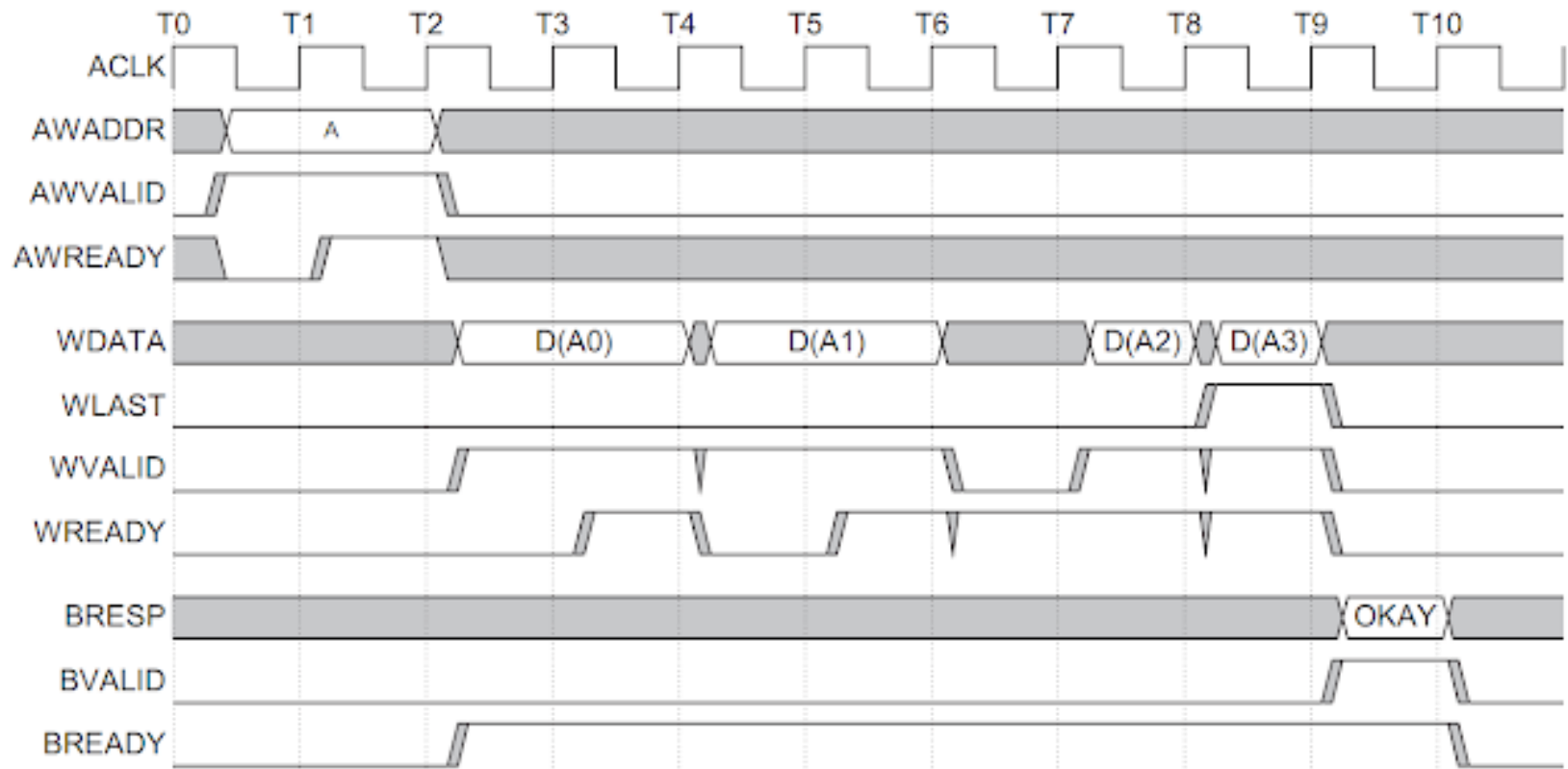


Write Transaction

- Three channels for write process
 - ➔ Write Address Channel
 - ➔ Write Data Channel
 - ➔ Write Response Channel
- Each channel has valid and ready signal for HS process



AXI Write Transfer (Burst length = 4)



AXI ordering

- AXI protocol enables **out-of-order** transaction with **multiple outstanding** addresses.

In order and
outstanding=1

address

a1		a2		a3	
----	--	----	--	----	--

data

	d1		d2		d3
--	----	--	----	--	----

In order and
outstanding>1

address

a1	a2	a3			
----	----	----	--	--	--

data

	d1		d2		d3
--	----	--	----	--	----

Out of order and
outstanding>1

address

a1	a2	a3			
----	----	----	--	--	--

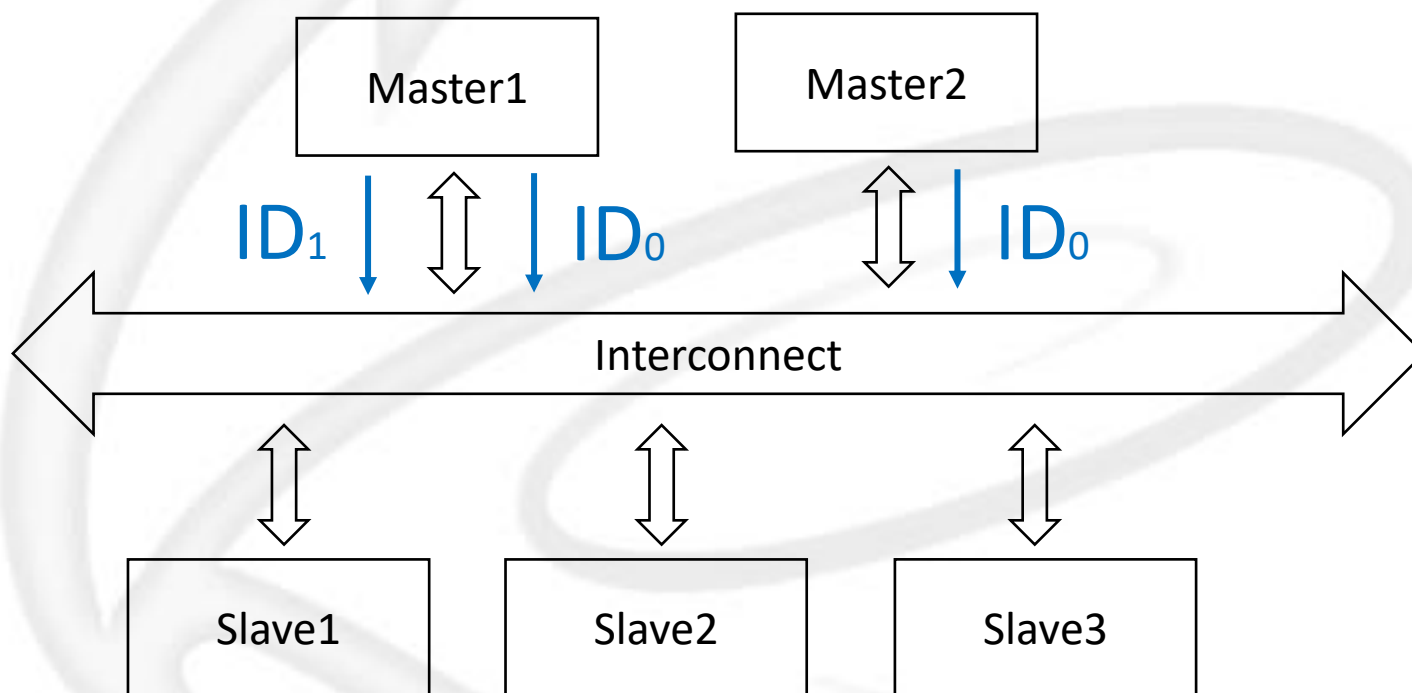
data

	d1		d3		d2
--	----	--	----	--	----

AXI ordering

- AXI has **ID signals** to support **out-of-order** and **outstanding** transactions.

IDs are only unique within the context of a single master

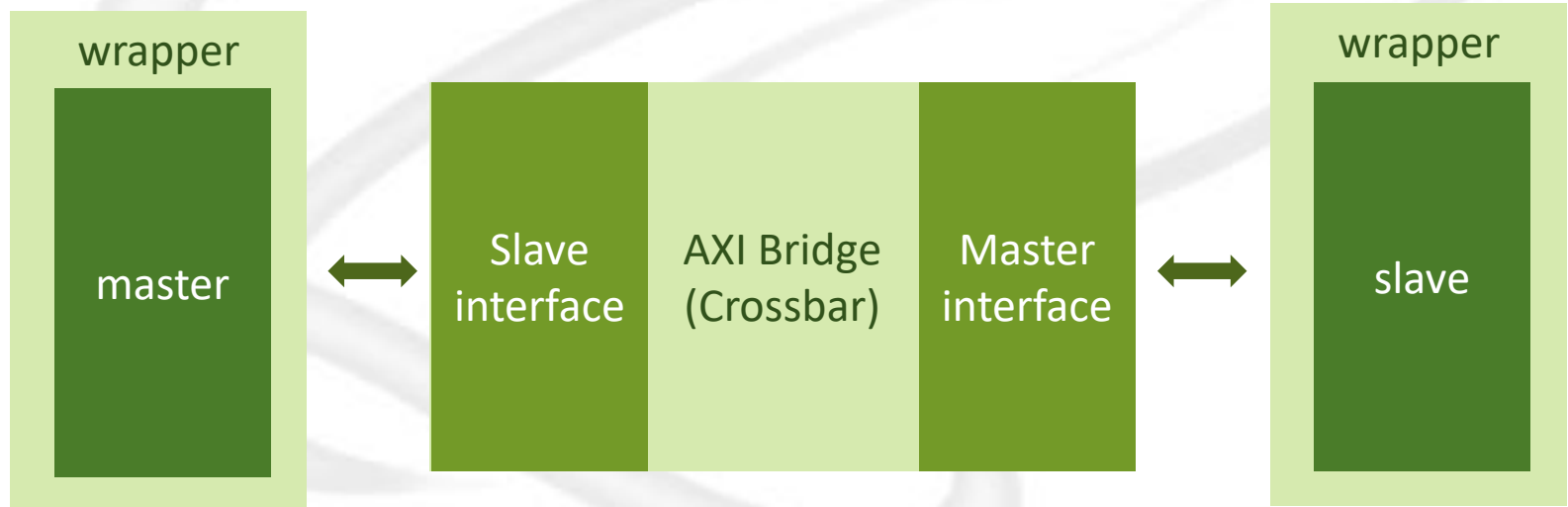


AXI ordering

- ❑ Transactions from different masters have no ordering restrictions.
- ❑ Transactions from same master, but different ID, have no ordering restrictions.
- ❑ The data for a sequence with same AWID or same ARID must complete in same order even if they are aiming at different slaves.
- ❑ There are no ordering restrictions with same AWID and ARID.
- ❑ For masters that only support single ordered interface, we can **tie the ID to a constant number**.

AXI Interconnect

- AXI Bridge
 - ➔ Crossbar
 - ➔ Slave Interface
 - ➔ Master Interface
- Master(s)
- Slave(s)



Port List in this Lab – Global Signals

Signal	Bits	Source	notes
ACLK	1	Clock source	Global clock signal
ARESETn	1	Reset source	Global reset signal, active LOW

--- 詳細的訊號說明請參照spec

Port List in this Lab – Read Address Signals

Signal	Bits	Source	notes
ARID	4/8	Master	Bits of master is 4 and for slave is 8, see A5-80 in spec
ARADDR	32	Master	
ARLEN	4	Master	Burst length.
ARSIZE	3	Master	Burst size.
ARBURST	2	Master	Burst type. Only need to implement INCR type.
ARVALID	1	Master	
ARREADY	1	Slave	

Port List in this Lab – Read Data Signals

Signal	Bits	Source	notes
RID	4/8	Slave	Bits of master is 4 and for slave is 8, see A5-80 in spec
RDATA	32	Slave	
RRESP	2	Slave	Read response. This signal indicates the status of the read transfer.
RLAST	1	Slave	Read last. This signal indicates the last transfer in a read burst.
RVALID	1	Slave	
RREADY	1	Master	

Port List in this Lab – Write Address Signals

Signal	Bits	Source	notes
AWID	4/8	Master	Bits of master is 4 and for slave is 8, see A5-80 in spec
AWADDR	32	Master	
AWLEN	4	Master	Burst length.
AWSIZE	3	Master	Burst size.
AWBURST	2	Master	Burst type. Only need to implement INCR type.
AWVALID	1	Master	
AWREADY	1	Slave	

Port List in this Lab – Write Data Signals

Signal	Bits	Source	notes
WDATA	32	Master	
WSTRB	4	Master	Write strobes. 4 bits because $32/8 = 4$
WLAST	1	Master	Write last. This signal indicates the last transfer in a write burst.
WVALID	1	Master	
WREADY	1	Slave	

Port List in this Lab – Write Response Signals

Signal	Bits	Source	notes
BID	4/8	Slave	Bits of master is 4 and for slave is 8, see A5-80 in spec
BRESP	4	Slave	Write response. This signal indicates the status of the write transaction.
BVALID	1	Slave	
BREADY	1	Slave	

Specification (1/2)

- ❑ Master : Single Transfer.
 - ➔ Burst length = 1
- ❑ Bridge and Slave : Burst transfer.
 - ➔ Burst length up to 4
- ❑ 2 masters and 2 slaves

Specification (2/2)

□ Slave

- ➔ Slave 1: 0x0000_0000 – 0x0000_ffff
- ➔ Slave 2: 0x0001_0000 – 0x0001_ffff
- ➔ Default slave: 0x0002_0000 – 0xffff_ffff

□ Default slave

- ➔ Response ERROR when masters access (RESP == DECERR) it

□ Default master

- ➔ Always execute, valid = 0 in the transfer
- ➔ You don't need to create a default master module

Module

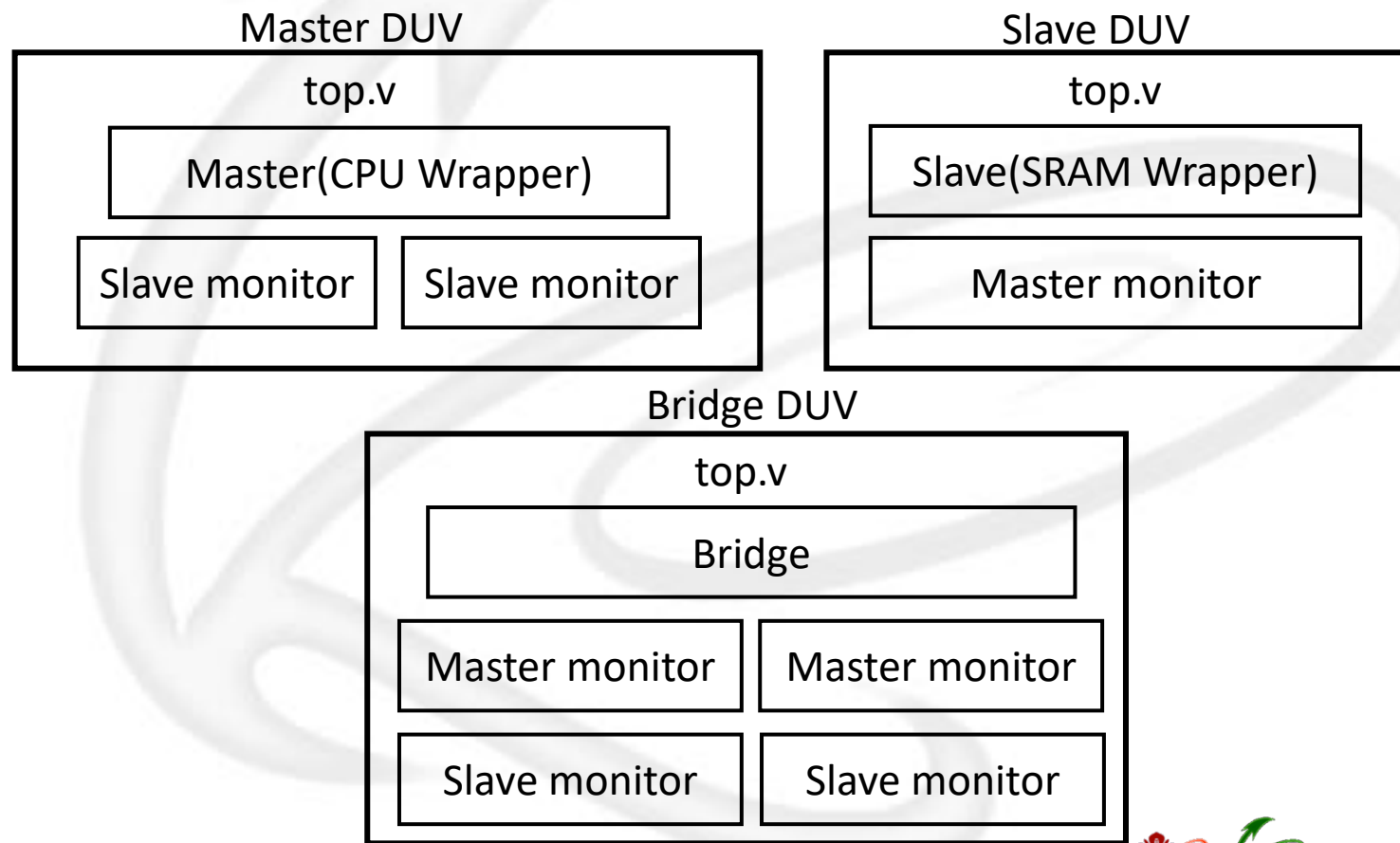
- ❑ Module names and module ports of AXI are defined. **DO NOT** modify them. The definition of Module ports can be checked in AXI spec.
- ❑ Ignore the signals that are not in this homework requirement.

Problem 1


作業驗證說明

Verification(1/3)

- 利用JasperGold的Verification IP分別去驗證CPU Wrapper, AXI, SRAM Wrapper



Verification(2/3)

- ❑ 請不要更改.tcl
- ❑ 在驗證MASTER DUV的時候，可能會出現前提不成立的狀況  (如：CPU 永遠READY)，需在報告中解釋原因，若為不合理原因則應修改設計
- ❑ Assertion應全數通過才算完整
- ❑ 請勿更動top.v的parameter

Verification(3/3)

Table B-1: Simulation commands

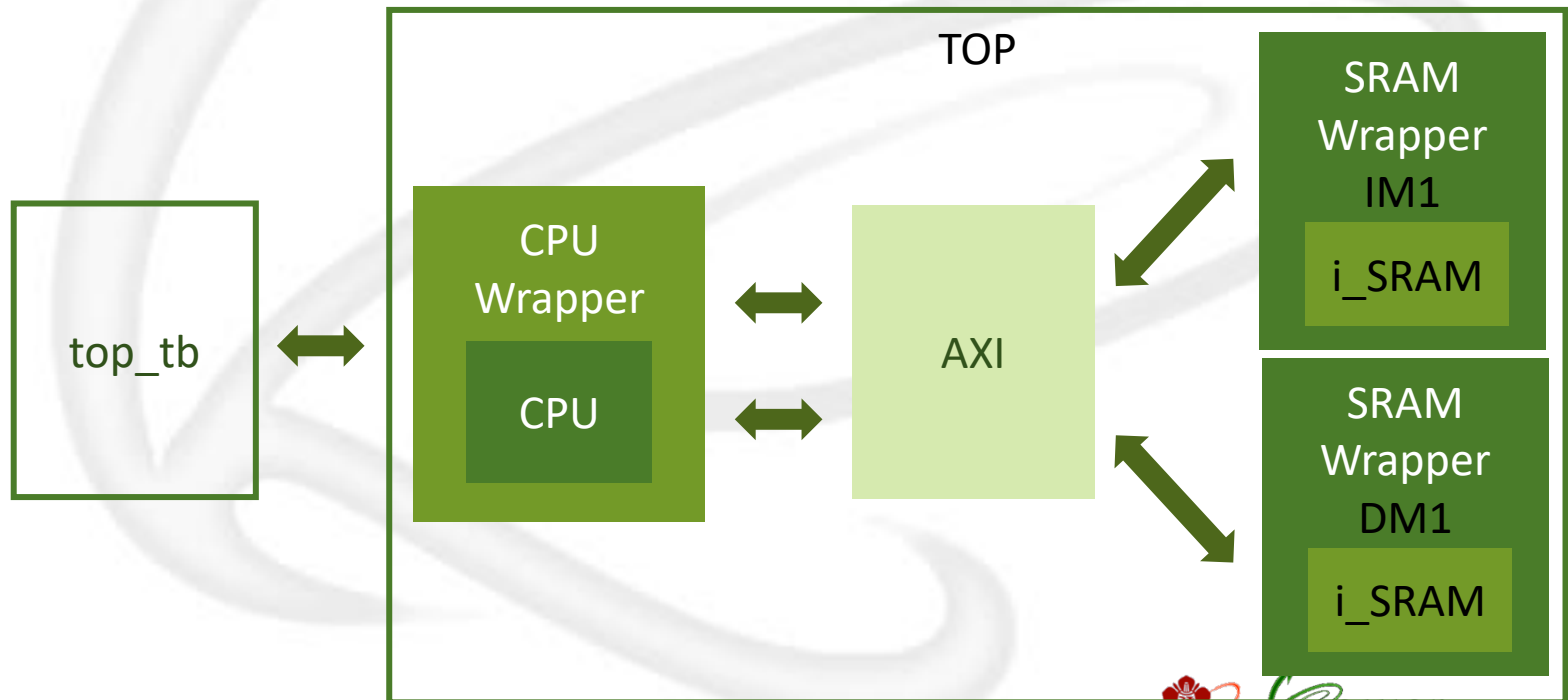
Situation	Command
Run JasperGold VIP on AXI bridge without file pollution (RTL only)	<code>make vip_b</code>
Run JasperGold VIP on AXI master without file pollution (RTL only)	<code>make vip_m</code>
Run JasperGold VIP on AXI slave without file pollution (RTL only)	<code>make vip_s</code>
Delete built 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>

Problem 2

作業驗證說明

Architecture

- CPU Wrapper
 - ➔ Instruction fetch (read only interface)
 - ➔ Load or store
- AXI
- SRAM Wrapper



Program

- prog0
 - ➔ 測試45個instruction (助教提供)
- prog1
 - ➔ Sort Algorithm
- Prog2
 - ➔ 不使用MUL/MUL[[S]U] instructions 完成Multiplication (助教提供)
- prog3
 - ➔ Greatest common divisor
- prog4
 - ➔ 使用階乘c code測試rdinstret, rdinstreth, rdcycle, rdcycleh(助教提供)
- Prog5
 - ➔ 使用MUL/MUL[[S]U] instructions實現Multiplication (助教提供)
- prog6
 - ➔ 使用floating point instructions進行運算(助教提供)

Specification

- ❑ Don't modify any timing constraint except clock period in *DC.sdc*. **Maximum clock period is 10 ns.**
- ❑ Design the master wrapper between CPU and AXI.
 - ➔ Transfer Signals between CPU and AXI
- ❑ Modify SRAM_wrapper to be compatible with AXI.
- ❑ CPU has two masters
 - ➔ Instruction
 - ➔ Data
- ❑ IM (Slave 1)
 - ➔ 0x0000_0000 – 0x0000_ffff
- ❑ DM (Slave 2)
 - ➔ 0x0001_0000 – 0x0001_ffff

Module (1/2)

- 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
RTL	SRAM_wrapper.sv	SRAM_wrapper	IM1	
RTL	SRAM_wrapper.sv	SRAM_wrapper	DM1	
RTL	SRAM_rtl.sv	TS1N16ADFPC LLLVTA512X45 M4SWSHOD	i_SRAM	
Gate-Level	TS1N16ADFPCLL LVTA512X45M4S WSHOD.sv	TS1N16ADFPC LLLVTA512X45 M4SWSHOD	i_SRAM	

- 以上 module name 助教均已提供或已定義好，請勿任意更改，以免 testbench 抓不到正確的名稱

Module (2/2)

- Module port須符合下表要求(同HW1)

Module	Specifications			
	Name	Signal	Bits	Function explanation
top	clk	input	1	System clock
	rst	input	1	System reset (active high)
TS1N16ADFPCL LLVTA512X45M 4SWSHOD	Memory Space			
	MEMORY	logic	32	Size: [512][32]

- 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

Simulation

Table B-1: Simulation commands (Partial)

Simulation Level	Command
Problem1	
RTL	<code>make rtl_all</code>
Post-synthesis (optional)	<code>make syn_all</code>

Table B-2: Makefile macros (Partial)

Situation	Command	Example
RTL simulation for progX	<code>make rtlX</code>	<code>make rtl0</code>
Post-synthesis simulation for progX	<code>make synX</code>	<code>make syn1</code>
Dump waveform (no array)	<code>make {rtlX,synX} FSDB=1</code>	<code>make rtl2 FSDB=1</code>
Dump waveform (with array)	<code>make {rtlX,synX} FSDB=2</code>	<code>make syn3 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>	
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>	

作業繳交注意事項



Report

- 請使用附在檔案內的Submission Cover
- 請勿將code貼在.docx內
 - ➔ 請將.sv包在壓縮檔內，不可截圖於.docx中
- 需要Summary及Lessons learned

繳交檔案 (1/2)

- 依照檔案結構壓縮成 “.tar” 格式
 - ➔ 在Homework主資料夾(N260XXXXX)使用**make tar**產生的tar檔即可符合要求
- 檔案結構請依照作業說明
- 請**勿**附上檔案結構內未要求繳交的檔案，斟酌扣分
 - ➔ 在Homework主資料夾(N260XXXXX)使用**make clean**可刪除不必要的檔案，但**仍需再確認**是否有多餘檔案沒有刪除
- 請務必確認繳交檔案可以在SoC實驗室的工作站下compile，且功能正常
- 無法compile將直接以0分計算
- 請勿使用generator產生code再修改
- 禁止抄襲

繳交檔案 (2/2)

- 請在合成資料夾(syn)中留下area_rpt.txt檔案

```

*****
Report : area
Design : top
Version: P-2019.03-SP1-1
Date   : Wed Sep 18 02:27:33 2024
*****

Library(s) Used:

  N16ADFP_StdCellss0p72vm40c (File: /usr/cad/CBDK/Executable_Package/Collaterals/IP/stdcell/N16ADFP_StdCell/NLDM/N16ADFP_StdCellss0p72vm40c.db)
  SRAM_ss0p72v0p72vm40c_100a (File: /home/user2/ms112/frank112/vsd_2024_lab2/sim/SRAM/SRAM_ss0p72v0p72vm40c_100a.db)

Number of ports:          6148
Number of nets:           24922
Number of cells:          18445
Number of combinational cells: 15526
Number of sequential cells:  2866
Number of macros/black boxes: 2
Number of buf/inv:         1686
Number of references:      8

Combinational area:        6000.013588
Buf/Inv area:              307.255689
Noncombinational area:     2847.623100
Macro/Black Box area:      9082.750000
Net Interconnect area:     undefined (Wire load has zero net area)

Total cell area:           17930.386688
Total area:                under the
1

```

```

syn (Your synthesized code and area file)
├── top_syn.v
├── top_syn.sdf
└── area_rpt.txt
vip (JasperGold ABVIP files)
├── bridge_duv (verify for AXI bridge)
│   ├── jg.f (You shouldn't modify it)
│   └── top.v (top module for AXI bridge verification with ABVIP)
├── master_duv (verify for AXI master)
│   ├── jg.f (You shouldn't modify it)
│   └── top.v (top module for AXI master verification with ABVIP)
└── slave_duv (verify for AXI slave)
    ├── jg.f (You shouldn't modify it)
    └── top.v (top module for AXI slave verification with ABVIP)

```

檔案結構 (1/2)

- N26XXXXXX.docx
→ Your report file
- src
→ Your source code (*.sv)
- include
→ Your definition code (*.sv)
- StudentID
→ Specify your Student ID number
- sim/CYCLE
→ Specify your clock cycle time
- sim/MAX
→ Specify max clock cycle number

- *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)
 - *Makefile* (You shouldn't modify it)
 - *src* (Your RTL code with sv format)
 - *top.sv*
 - *SRAM_wrapper.sv*
 - Other submodules (*.sv)
 - *AXI*
 - *AXI.sv*
 - Submodules of AXI (*.sv)
 - *include* (Your RTL definition with svh format)
 - *AXI_def.svh*
 - Definition files (*.svh)
 - *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)
 - *CYCLE* (Specify your clock cycle time in this file)
 - *MAX* (Specify max clock cycle number in this file)
 - *SRAM* (SRAM libraries and behavior models)
 - Library files (*.lib, *.db, *.lef or *.gds)
 - *TS1N16ADFPCLLLVTA512X45M4SWSHOD.sv* (SRAM behavior model)
 - *SRAM_rtl.sv* (SRAM RTL model)

檔案結構 (2/2)

- sim/prog0 、 prog2 、 prog4 、 prog5 、 prog6
→ Don't modify contents
- sim/prog1 、 prog3
→ main.S
→ main.c
◆ Submit one of these
- Don't modify Makefile

```

└─ syn (Your synthesized code and area file)
    └─ top_syn.v
    └─ top_syn.sdf
    └─ area_rpt.txt
└─ vip (JasperGold ABVIP files)
    └─ bridge_duv (verify for AXI bridge)
        └─ jg.f (You shouldn't modify it)
        └─ top.v (top module for AXI bridge verification with ABVIP)
    └─ master_duv (verify for AXI master)
        └─ jg.f (You shouldn't modify it)
        └─ top.v (top module for AXI master verification with ABVIP)
    └─ slave_duv (verify for AXI slave)
        └─ jg.f (You shouldn't modify it)
        └─ top.v (top module for AXI slave verification with ABVIP)
    
```

Advanced 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.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)

prog2 (Subfolder for Program 2)

- Makefile (Compile and generate memory content)
- 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)
- 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)

prog4 (Subfolder for Program 4)

- Makefile (Compile and generate memory content)
- 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)

prog5 (Subfolder for Program 5)

- Makefile (Compile and generate memory content)
- 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)

prog6 (Subfolder for Program 6)

- Makefile (Compile and generate memory content)
- 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)

Level1 Fall 2024

繳交期限

- 2024/11/06 (三) 15:00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

注意事項

- 本次作業合成的部分有計入PA(Performance & Area)，表現較為優異者將有較高的評分
- 作業部分有任何問題請在moodle上的作業討論區發問並可參考其他人是否有類似的問題，助教信箱恕不回覆。
- 在Script/DC.sdc中，可以更改clk period範圍，請注意最大的接受值為10.0，超過此範圍者恕不納入計分。此外，若有更改預設clock period者，請務必更改set_input_delay -max至1/2 clk period，以利後續作業的合成及模擬。
- 如有需要可以更改synthesis.tcl的合成指令來達到timing要求，但不可更改產生合成檔案的指令

```
#area report  
report_area -nosplit > ../syn/area_rpt.txt  
#report power  
report_power -analysis_effort low > ../syn/power_rpt.txt
```

```
# Save synthesized file  
write -hierarchy -format verilog -output {../syn/top_syn.v}  
#write_sdf -version 1.0 -context verilog {../syn/top_syn.sdf}  
write_sdf -version 3.0 -context verilog {../syn/top_syn.sdf}
```

- 本次作業需在SoC環境下模擬，請同學務必在SoC教室執行模擬，若是助教在評分作業時於SoC無法成功模擬則以0分計算。

**Thanks for your participation and
attendance !!**