NCTU-EE IC LAB – Fall 2019

Midterm Project

Release/Deadline : 2019.11.06 12:00 / 2019.11.25 12:00

Design: Mail Transfer Center

Data Preparation

- Extract test data from TA's directory:

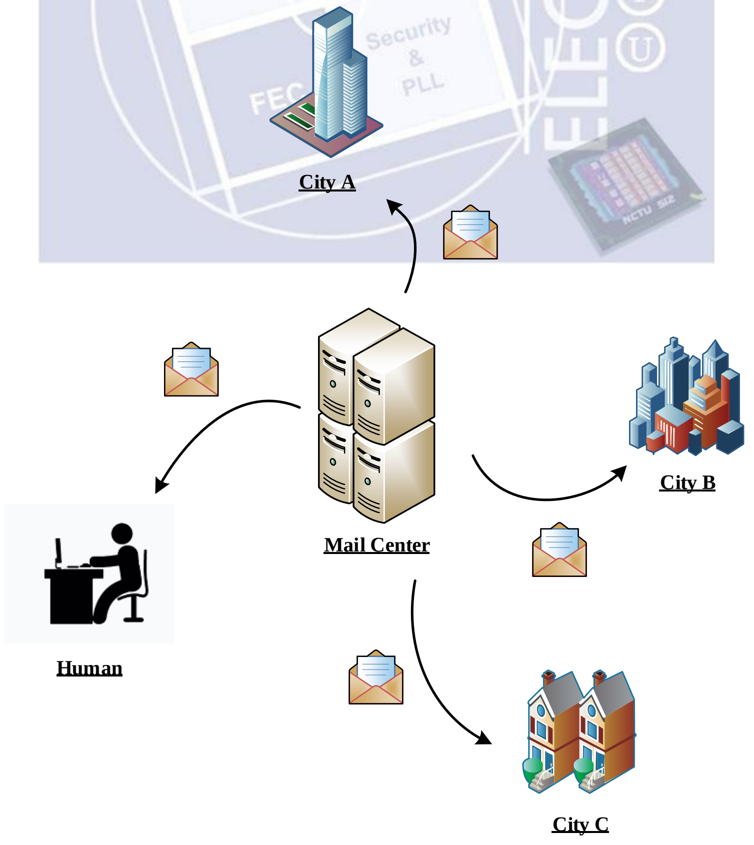
  % tar xvf ~iclabta01/Mid.tar

- The extracted Lab director contains:

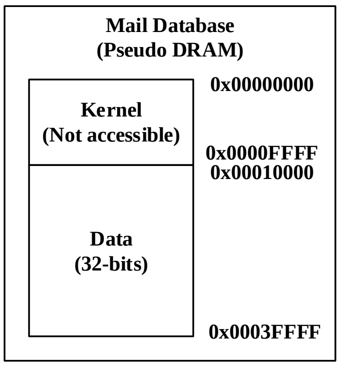
  a. Mid/ : Midterm project

Design Description

There is a government secret mail system. Any mail should be sent to the mail center and check it is legal or not before it arrives at recipient. The mails will be put in a big database. The mail center should deliver the mails, which are in the database, to the target city. In this country, there are three cities, which are city A, city B and city C. If there exits any suspect content, the people who work for government will check it.



Mail Database



32-bits			
8-bits	6-bits	14-bits	4-bits

Example

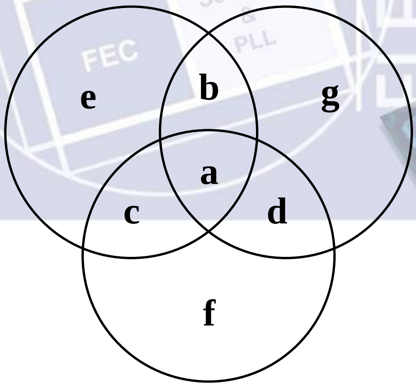
Address (hexadecimal)	**0x00001003**	**0x00001002**	**0x00001001**	**0x00001000**
Data(decimal)	0100_0111	0001_0100	0000_0000	0100_1111

8-bits	**0100_0111**	14-bits	**00_0000_0000_0100**
6-bits	**00_0101**	4-bits	**1111 (0xF)**

32-bits			
8-bits	6-bits	14-bits	4-bits

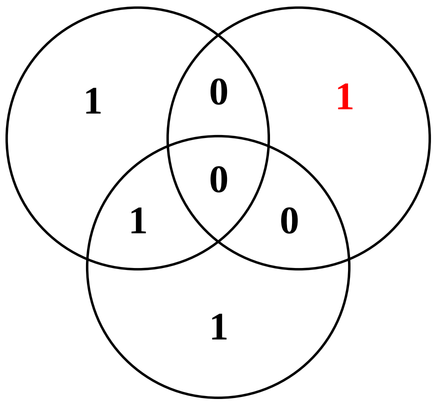
8-bits	**: Hamming code**
6-bits	**: Destination**
14-bits	**: Address**
4-bits	**: Content**

➢ **Hamming code**



s	**a**	**b**	**c**	**d**	**e**	**f**	**g**

Example : codeword = 001_0111



correct codeword = 001_0110

If original codeword is not correct, **s** should be set to high. (1001_0110 for example). In this case, you need to write back to mail database. The first bit, **s**, means this mail contains suspect content. **The suspect mail will be blocked by the government.**

- **Destination**

  In general, the mail will be encrypted before sending it. So, we need to decode it and find the destination of the mail. We can consider it as D[5:0].

Operation	D[5:4]	(bitwise)
XOR	00	D[3:2] ⊕ D[1:0]
AND	01	D[3:2] · D[1:0]
OR	10	D[3:2] +D[1:0]
NOT	11	! D[1:0]

Result	City
00	A
01	B
10、11	C

Example: D[5:0] = 10_1001, result = 10 + 01 = 11. In this case, the destination is city C.

- **Address (14-bits)**

  Address means the memory address of the destination. The address can be divisible by 4, e.g. 0x00000, 0x00004, 0x0000B etc.
- **Content (4-bits)**

  You need to store the content in target DRAM memory. (0x0~0xF)

Inputs

I/O	Signal Name	Description
Input	clk	Positive edge trigger clock within clock period 20.0ns
Input	rst_n	Asynchronous reset active low reset
Input	start_i	You are asked to transfer 256 mails from mail database when **start_i** is high. **start_i** will be high for only one cycle. The next input pattern will come in 1~3 cycles after **done_o** falls.

- ➢ All inputs will be changed at clock ***negative*** edge.
- ➢ There is ***only 1 reset*** before the first pattern, thus, your design must be able to reset automatically.

Outputs

I/O	Signal Name	Description
Output	done_o	**done_o** should be low after initial reset. **done_o** should not be raised when **start_i** is high. After **done_o is high, Pattern will check the correctness of the value inside DRAM.**

The test pattern will check whether your output sequence is correct or not at clock ***negative*** edge.

Pattern

- ➢ Update DRAM data
  00_TESTBED/DRAM/dram_file.dat
  Hint: You can refer lecture note for Lab03 about file input.
- ➢ DRAM Raad/Write Latency
  Modify **DRAM_R_LAT**, **DRAM_W_LAT** in the DRAM. You can set DRAM_R_LAT = 0 & DRAM_W_LAT = 0 to speed up the simulation time. But for demo, TA will set **DRAM_R_LAT = 90** and **DRAM_W_LAT = 100.**

AXI 4 Interface (Connected with DRAM in Pattern)

TA has already declared for you in MATRIX_SYS.v

AXI-4 signal name (lower case) + _m_inf (suffix)

Write Address Channel

Signal	Source	Description
AWID[3:0]	Master	Write address ID. This signal is the identification tag for the write address group of signals. (In this project, we only use this to recognize master, reordering method is not supported)
AWADDR[31:0]	Master	Write address. The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst transaction.
AWLEN[7:0]	Master	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
AWSIZE[2:0]	Master	Burst size. This signal indicates the size of each transfer in the burst. (We only support 3b'010 which is 4 Bytes (matched with Data Bus-width) in each transfer)
AWBURST[1:0]	Master	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. (only INCR support in this Project)
AWVALID	Master	Write address valid. This signal indicates that valid write address and control information are available: 1 = address and control information available 0 = address and control information not available. The address and control information remain stable until the address acknowledge signal, **AWREADY**, goes HIGH.
AWREADY	Slave	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready 0 = slave not ready.

Write Data Channel

Signal	Source	Description
WDATA	Master	Write data. The write data bus can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide. (This project only support 32 bit data width: WDATA[31:0])
WLAST	Master	Write last. This signal indicates the last transfer in a write burst.
WVALID	Master	Write valid. This signal indicates that valid write data and strobes are available: 1 = write data and strobes available 0 = write data and strobes not available.
WREADY	Slave	Write ready. This signal indicates that the slave can accept the write data: 1 = slave ready 0 = slave not ready.

Write Response Channel

Signal	Source	Description
BID[3:0]	Slave	Response ID. The identification tag of the write response. The **BID** value must match the **AWID** value of the write transaction to which the slave is responding.
BRESP[1:0]	Slave	Write response. This signal indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR. (In this project we only issue OKAY)
BVALID	Slave	Write response valid. This signal indicates that a valid write response is available: 1 = write response available. 0 = write response not available.
BREADY	Master	Response ready. This signal indicates that the master can accept the response information. 1 = master ready. 0 = master not ready.

Read Address Channel

Signal	Source	Description
ARID[3:0]	Master	Read address ID. This signal is the identification tag for the read address group of signals. (In this project, we only use this to recognize master, reordering method is not supported)
ARADDR[31:0]	Master	Read address. The read address gives the address of the first transfer in a read burst transaction.
ARLEN[7:0]	Master	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
ARSIZE[2:0]	Master	Burst size. This signal indicates the size of each transfer in the burst. (We only support 3b'010 which is 4 Bytes (matched with Data Bus-width) in each transfer)
ARBURST[1:0]	Master	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. (only INCR: 2b'01 support in this Project)
ARVALID	Master	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledge signal, **ARREADY**, is high. 1 = address and control information valid 0 = address and control information not valid.
ARREADY	Slave	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready 0 = slave not ready.

Read Data Channel

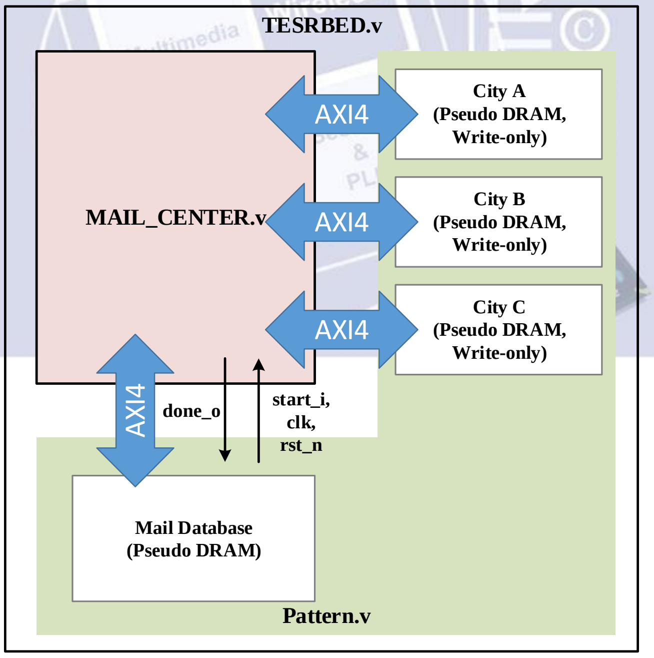
Signal	Source	Description
RID[3:0]	Slave	Read ID tag. This signal is the ID tag of the read data group of signals. The **RID** value is generated by the slave and must match the **ARID** value of the read transaction to which it is responding.
RDATA	Slave	Read data. (This project only support 32 bit data width: RDATA[31:0])
RRESP[1:0]	Slave	Read response. This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR. (In this project we only issue OKAY)
RLAST	Slave	Read last. This signal indicates the last transfer in a read burst.
RVALID	Slave	Read valid. This signal indicates that the required read data is available and the read transfer can complete: 1 = read data available 0 = read data not available.
RREADY	Master	Read ready. This signal indicates that the master can accept the read data and response information: 1= master ready 0 = master not ready.

Specifications

1. Top module name: **MAIL_CENTER** (top file name: **MAIL_CENTER.v**)
2. It is **asynchronous** reset and **active-low** architecture. **rst_n** would active once at the beginning.
3. All the data are in **unsigned** format.
4. The latency of your design in each pattern should not be larger than **100,000** cycles.(with DRAM latency mode **off**)
5. The maximum clock period is set to **20ns**, and **you can determine the clock period by yourself**.
6. The input delay and the output delay should be **half** of the clock period. For example, if clock period is 10ns, the input delay and the output delay will be 5ns.

7. The output loading is set to **0.05**. The wire load is **top** and target library is **slow.db**
8. The synthesis result of data type cannot include any **LATCH** (in syn.log).
9. In this project, you should **modify the syn.tcl (e.g. link library for your memory.db) by yourself**. **compile_ultra** will be used to synthesis. Since memories are used in this project, the syn.tcl in Lab05 may be a good reference one.
10. You **CANNOT PASS** the demo if there are **timing violation** messages in the gate level simulation WITHOUT notimingchecks option
11. No **ERROR** is allowed in every simulation/synthesis.
12. The **size of SRAM Memory Block** should between **256~2048 Bytes**, the **number of SRAM is not limited.**
13. **You must use memory in your design or you wouldn't get any point in this project.**
14. The maximum area should be smaller than **2,000,000.**
15. If there is no mail, the content should be **zero** for DRAM_A~C_city.
16. **The mail will be overwritten by the new one.**
17. **There is no kernel space in DRAM_A~C_city.**

Block Diagram


TESRBED.v
MAIL_CENTER.v
AXI4
City A (Pseudo DRAM, Write-only)
AXI4
City B (Pseudo DRAM, Write-only)
AXI4
City C (Pseudo DRAM, Write-only)
AXI4
done_o
start_i, clk, rst_n
Mail Database (Pseudo DRAM)
Pattern.v

Grading Policy

1. The performance is determined by the area and the latency of your design. The smaller the performance index is, the higher score you can get.

   Functionality: 75%

   Performance: (Total Latency)$^{1.2}$ x Area x Clock Period : 25%

(in this performance metric latency means cycle count)

Note

1. Please upload the following files on new e3 platform before **12:00 noon** on **Nov. 25**.

   RTL design : **MAIL_CENTER_iclab??.v**, (?? is your account number)

   Memory File : **MEMORY_NAME_iclab??.v**

   MEMORY_NAME_iclab??.db

   (?? is your account number).

   Tcl (partially): **file_list_iclab??.f** **syn_iclab??.tcl**

   Example:

   Given your memory name is RA1SH512, and your submitted memory file RA1SH512_iclab99.v

   in file_list_iclab99.f

   ```
   ../04_MEM/RA1SH512.v
   ```

   Clock period: **your_clock_iclab??.txt** (e.g. 5_iclab99.txt)

   Note that you can provide **multiple** memory specs in this lab. If the uploaded files violating the naming rule, you will get **5 deduct points**.

2. Template folders and reference commands:

   01_RTL/ (for RTL simulation) **./01_run**

   02_SYN/ (for Synthesis) **./01_run_dc**

   Remember modify .tcl to fit your memory db name

   (Check the design which contains **Latch** and **Error** or not in **syn.log**)

   (Check the design's timing in /Report/MAIL_CENTER.timing to see if the slack is **MET**)

   03_GATE_SIM/ (Gate Level simulation) **./01_run**

   You can key in **./09_clean_up** to clear all log files and dump files in each folder

   04_MEM/ (Memory location)

   You should generate your memories and put the required files (.v and .db) here

Hint

1. In this project you may use multiple kinds of memories, thus you have to provide multiple netlists and libraries for the simulation and the synthesis.

For example, if you use two memories, you have to provide two .v and .db file. Here provides a simple flow to perform:

Login to the Memory Compiler Server :

Copy the template memory generate folder :

Go to the folder and execute the shell script.

Argument of script: [name] [number of word] [number of bit] [mux type] [frequency]

Frequency could be any value larger than 0 (We suggest you set to 1)

Example:

Account Name: iclab99

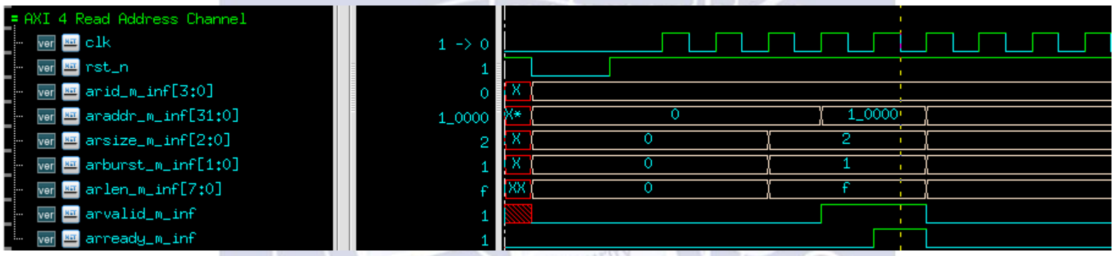
Spec: name: SRAM128W64B number of word:128

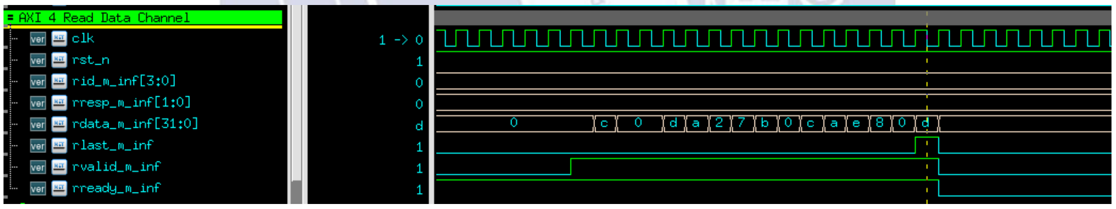
number of bit : 64 mux type :4 frequency : 1

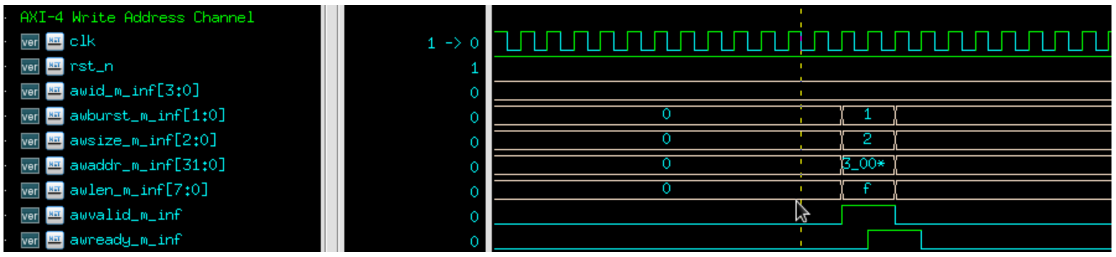
```
ssh mem@ee08.ee.nctu.edu.tw
cp  -r  mid_mem_template  iclab99_mid
cd iclab99_mid
./01_mem_gen.sh  SRAM128W64B  128  64  4  1
```

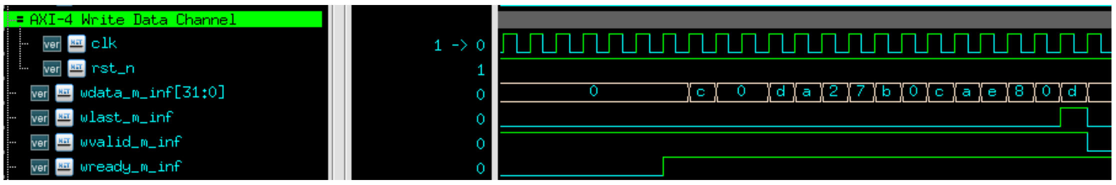
AXI example waveform:

Read Transaction

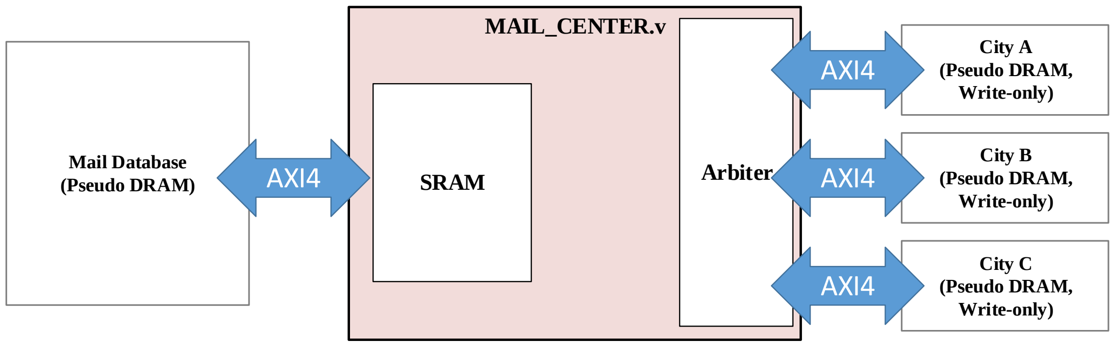








Hint:


MAIL_CENTER.v
Mail Database (Pseudo DRAM)
AXI4
SRAM
Arbiter
AXI4
City A (Pseudo DRAM, Write-only)
AXI4
City B (Pseudo DRAM, Write-only)
AXI4
City C (Pseudo DRAM, Write-only)