frNCTU-EE IC LAB – Fall2019

Lab09 Exercise–Design and Verification Using SystemVerilog

Design: Coin Transaction System

Data Preparation

1. Extract test data from TA's directory:

    % tar xvf ~iclabta01/Lab09.tar

2. The extracted LAB directory contains

    Exercise/: your exercise

Design Description

In this lab, we are going to bulid a Coin Transaction System based on AXI Lite. This system allows users to transfer money, make a deposit, change ID password, or simply check balance of their account.

Operation

- **Log in**
  step1: enter ID, ID = { bank_ID, account }
  step2: enter password

- **Transfer money**
  step1: enter transfer account
  step2: enter the amount, transfer fee = (**amount** x $^{1}/_{16}$)
  step3: show balance

- **Make a deposit**
  step1: enter the amount
  step2: show balance

- **Change ID password**
  step1: enter new password

- **Check balance**
  step1: show balance

- **Exit**

ACTION		ERROR MESSAGE	
transfer money	4’b 0010	Out of money	4’b 0001
make a deposit	4’b 0011	Wrong password	4’b 0010
check balance	4’b 0101	Account not exist	4’b 0100
change password	4’b 0110	Invalid new password	4’b 1000
exit	4’b 0111		

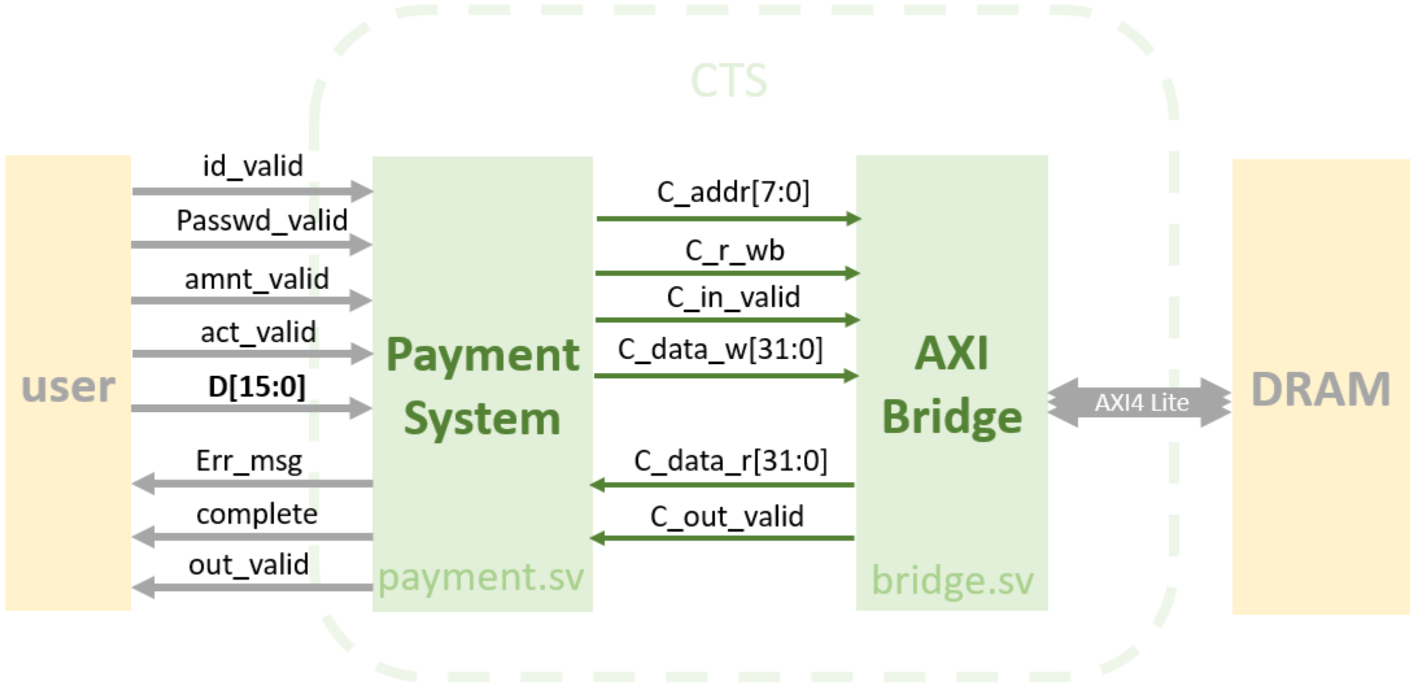
Following is some rules about the actions:

1. Only when the previous user has log out, the next user could log in.
2. If account not in use, the password will be 4’b0000.
3. If user log in with a non-exist account, show error message “ *Account not exist* ”.
4. If user log in with wrong password, show error message “ *Wrong password*”.
5. If user want to change password and enter new password = 4’b0000, show error message “ *Invalid new password*”.
6. If user want to transfer money, and the amount is more than his/her deposit, show error message “ *Out of money* ”.
7. If any error message is triggered, the action won’t be executed, and the system status should remain the same.
8. If someone wants to transfer money to account **under different bank**, the transfer action needs transfer fee = $\textbf{amount} \times {}^{1}/_{16}$, taken from current user account. Round down if transfer fee is not integer.

All the other rules can be referred to examples in Lab09_Exercise_note.pdf.

Design block diagram

Here is the rule of this design:



1. After **rst_n**, all the output signals should be set to 0.
2. Please initialize DRAM at beginning. (please refer to Lab09_exercise_note.pdf)
3. The pattern will insert the pattern using **4 valid signal + 1 data signal** :

id_valid	High when user enter ID
passwd_valid	High when user enter current password or new password
amnt_valid	High when user enter transfer amount
act_valid	High when user enter action

D[15:0]	The content of current input
	Ex:
	account = { 8’b0, XXXXXXXX }
	passwd = { 12’b0, XXXX }

4. You need to raise **out_valid** when your design has been done for the input supply operation.
5. We have total 16 banks and each bank could contain at most 16 accounts.
   Id = { Bank_ID, Acc.No.}
6. In DRAM, we don’t directly store the password for safety issue.
   We store PW xor ID || Bank ID in stead.
   Ex: ID = {4’b 1111, 4’b 0011} PW = 4’b1000

0011^1000 = 1011, we store **{4'b1011, 4'b1111}** in DRAM.

7. We get data (ex: balance, encrypted password) via AXI Lite protocol, and AXI Lite can only transfer 32 bits of data at one time.
8. Since the password we get from DRAM is encrypted, we need to do some inverse operation if we want to access the real password.

Input: Payment.sv

name	**type**	**from**	note
clk	1-bit	testbench	System clock
rst_n	1-bit	pattern	Asynchronous reset active low reset. Every output signal should be zero after **rst_n**.
id_valid	1-bit	pattern	High when user enter current login account or the account receives transfer.
passwd_valid	1-bit	pattern	High when user enter current password or new password.
amnt_valid	1-bit	pattern	High when user enter the amount of money to be transferred or deposited.
act_valid	1-bit	pattern	High when user enter action.
D[15:0]	16-bit	pattern	Represents the contents of current input.
C_out_valid	1-bit	Bridge	High when data from DRAM is ready.
C_data_r	32-bit	Bridge	The returned data from DRAM, including user ID, user encrypted password, and current balance

Output: Payment.sv

name	width	Send to	
out_valid	1-bit	pattern	Should set to high when your output is ready. **out_valid** will be high only one cycle.
err_msg	4-bit	pattern	Err_msg will be 4’0000 if operation is complete, else it needs to be corresponding value.
complete	1-bit	pattern	1’b1 : operation complete 1’b0 : some error occurred, so the status of DRAM won’t change.
out_balance	16-bit	pattern	Show balabce of current account, if action is transfer money, make a deposit or check balance.
C_addr	8-bit	bridge	C_Addr Indicates which address we want to access.
C_data_w	32-bit	bridge	The data to over right DRAM, including user ID, user encrypted password, and current balance
C_in_valid	1-bit	bridge	High when payment system is ready to communicate with bridge.
C_r_wb	1-bit	bridge	1’b1: Read DRAM. 1’b0: Write DRAM.

Input: Bridge.sv

name	type	from	
clk	1-bit	pattern	System clock
rst_n	1-bit	pattern	Asynchronous reset active low reset. Every output signal should be zero after **rst_n**.
C_addr	8-bit	payment	C_Addr Indicates which address we want to access.
C_data_w	32-bit	payment	The data to over right DRAM, including user ID, user encrypted password, and current balance
C_in_valid	1-bit	payment	High when payment system is ready to communicate with bridge.
C_r_wb	1-bit	payment	1'b1: Read DRAM. 1'b0: Write DRAM.
AR_READY	1-bit	DRAM	AXI Lite signal
R_VALID	1-bit	DRAM	AXI Lite signal
R_DATA	32-bit	DRAM	AXI Lite signal
AW_READY	1-bit	DRAM	AXI Lite signal
W_READY	1-bit	DRAM	AXI Lite signal
B_VALID	1-bit	DRAM	AXI Lite signal
B_RESP	2-bit	DRAM	AXI Lite signal

Output: Bridge.sv

name	width	Send to	
C_out_valid	1-bit	payment	High when data from DRAM is ready.
C_data_r	32-bit	payment	The returned data from DRAM, including user ID, user encrypted password, and current balance
AR_VALID	1-bit	DRAM	AXI Lite signal
AR_ADDR	17-bit	DRAM	AXI Lite signal
R_READY	1-bit	DRAM	AXI Lite signal
AW_VALID	1-bit	DRAM	AXI Lite signal
AW_ADDR	17-bit	DRAM	AXI Lite signal
W_VALID	1-bit	DRAM	AXI Lite signal
W_DATA	32-bit	DRAM	AXI Lite signal
B_READY	1-bit	DRAM	AXI Lite signal

Specifications

1. Top module name: **CTS** (File name: **bridge.sv** & **payment.sv**)
2. The type defined in Usertype_PKG.sv by TA should not be modified. But you are encouraged to defined new datatype if needed.
3. Using **asynchronous reset active low** architecture. All outputs should be zero after reset.
4. All outputs are synchronized at clock rising edge.
5. All input signal won't be unknown after reset.
6. Design limit and synthesis constraint:
    a. Maximum clock period is **15 ns**.
    b. You can change the clock period for better performance (unit is 0.5 ns).
    c. The input delay and output delay are **"0.5*clock period"**.
    d. The output load should be set to **0.05**
7. The synthesis result of data type cannot include any **LATCH**.
8. Cell library and Designware are in the same directory as previous exercises.
9. **rst_n** would be active once in the beginning. After that, any timing violation and glitch is not allowed
10. Your latency should be less than 1200 cycle in each operation .

Grading

- **Function: 70%**
- **Performance: (bridge_Area + payment_Area) x latency**

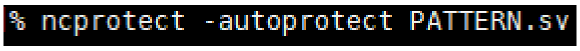
Note

- Upload your design on e3 before 12:00 at noon on 12/2
- Upload your design and package
  Usertype_PKG_iclabXX.sv
  iclabXX_clock_period.txt
  payment_iclabXX.sv
  bridge_iclabXX.sv

Reference Waveform

Please refer to Lab09_exercise_note.pdf

※ Since Lab10 is about pattern in SystemVerilog, so if you need to give your pattern to others, please use the following command to encrypt your code.



```
% ncprotect -autoprotect PATTERN.sv
```