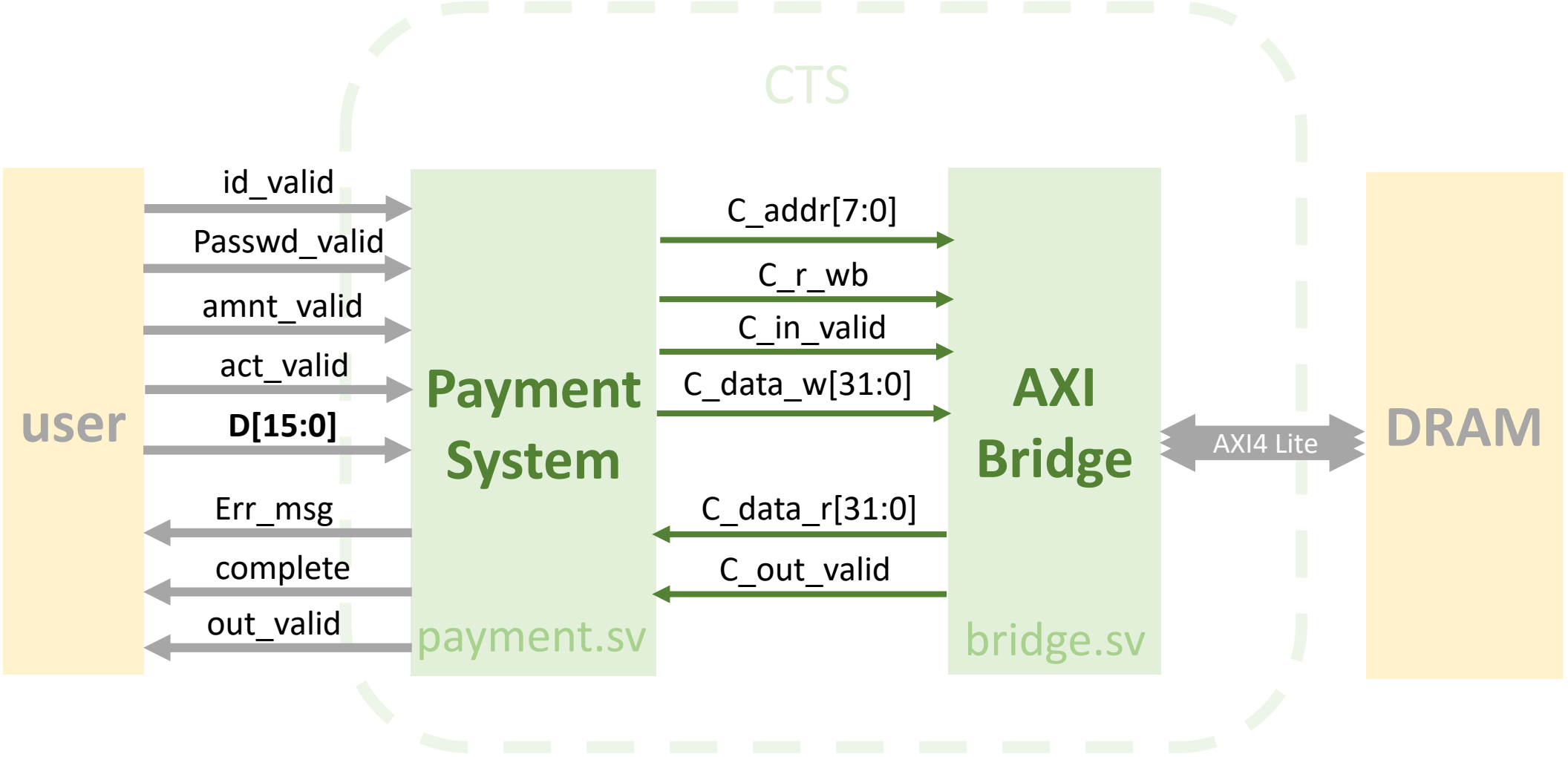
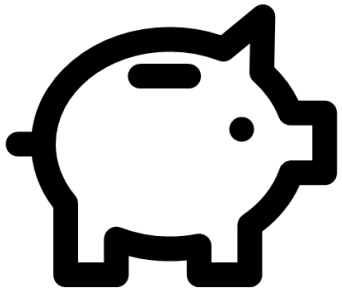

CTS
user
id_valid
Passwd_valid
amnt_valid
act_valid
D[15:0]
Err_msg
complete
out_valid
Payment System
payment.sv
C_addr[7:0]
C_r_wb
C_in_valid
C_data_w[31:0]
C_data_r[31:0]
C_out_valid
AXI Bridge
bridge.sv
AXI4 Lite
DRAM

Account order

Bank 0



Acct.No 0

Account order

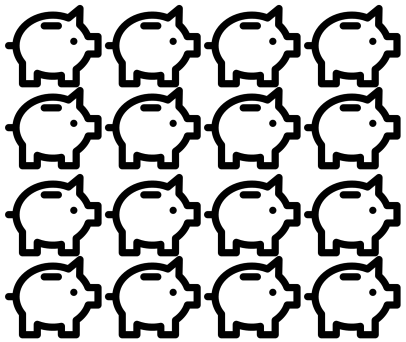
Bank 0

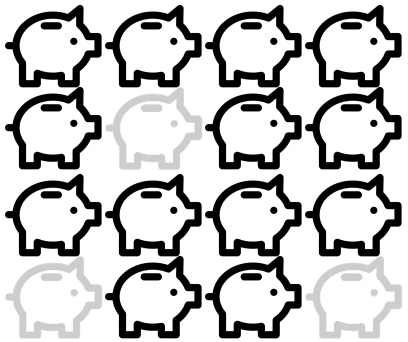
Bank 1

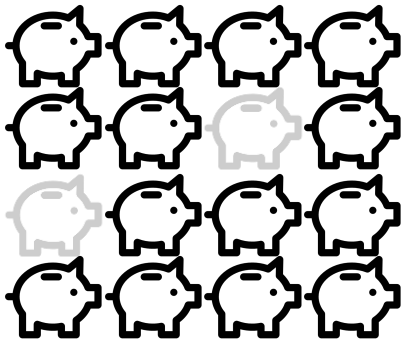
...

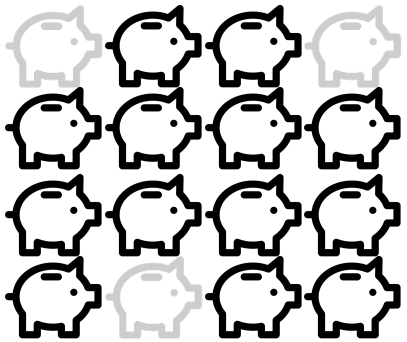
Bank 14

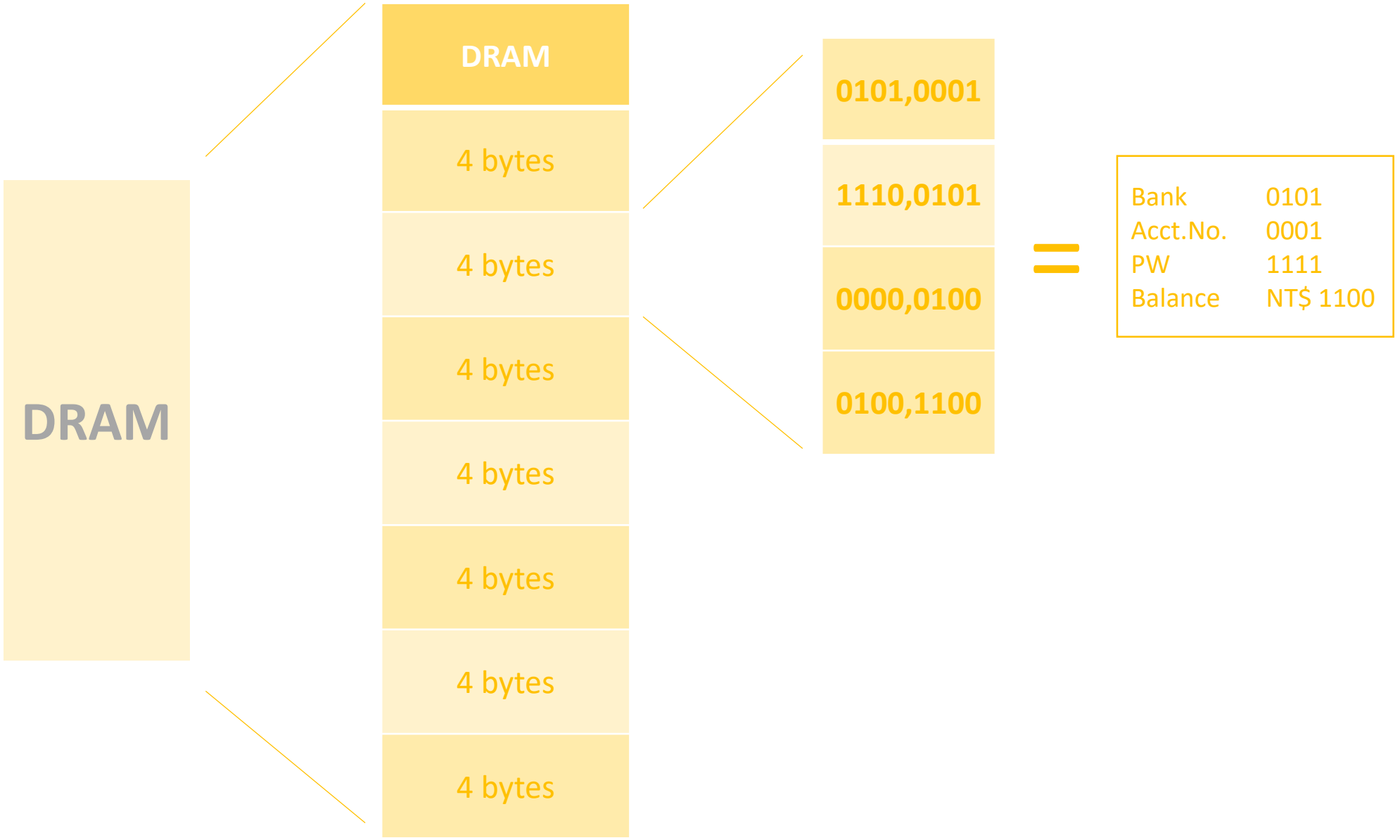
Bank 15










DRAM
DRAM
4 bytes
4 bytes
4 bytes
4 bytes
4 bytes
4 bytes
4 bytes
0101,0001
1110,0101
0000,0100
0100,1100
=
Bank 0101
Acct.No. 0001
PW 1111
Balance NT$ 1100

Dram file example

```
1   @10000
2   0 60 0 5
3   @10004
4   1 c0 89 a6
5   @10008
6   2 b0 27 7
7   @1000c
8   3 0 ed 8a
```

```
37  @10048
38  12 61 5b 35
39  @1004c
40  13 71 7a 23
41  @10050
42  14 e1 ad 50
43  @10054
44  15 31 e8 6e
```

```
505  @103f0
506  fc bf ee bf
507  @103f4
508  fd 4f 8c 37
509  @103f8
510  fe bf a3 db
511  @103fc
512  ff 2f 46 cc
```

```
43  @10054
44  15 31 e8 6e
```

=

Bank	1	0001
Acct.No.	5	0101
PW	6	0110
Encrypt pw	31	0011,0001
Balance	e86e	16’d 59502

Note

- You may use the following code to initialize your DRAM in your pattern program

.dat file location →
```
parameter DRAM_p_r = "../00_TESTBED/DRAM/dram.dat";
```

Declaration of dram reg array →
```
logic [7:0] golden_DRAM[(65536+256*4)-1:65536+0];
```

Initialize dram →
```
initial $readmemh(DRAM_p_r, golden_DRAM);
```

Example

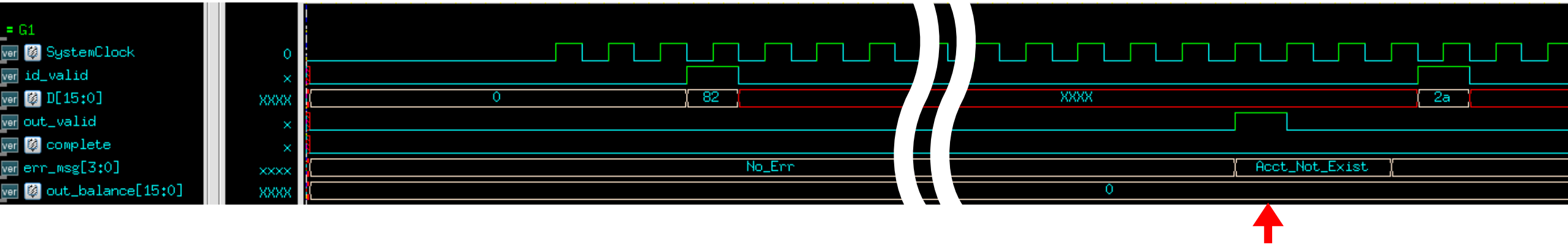
- Case 1 – user log in, but account not exist
- Case 2 – user log in, but password is wrong
- Case 3 – transfer money to unregistered account
- Case 4 – transfer too much money
- Case 5 – transfer money to acct under different bank
- Case 6 – make a deposit
- Case 7 – check balance
- Case 8 – change password successfully
- Case 9 – change password, but new password is illegal
- Case 10 – log in then directly exit
- Case 11 – after exit, next user log in

Bank = 4'h 8
Acct_no = 4'h 2
passwd = 4'b 1000
Balance = 16'h 8282

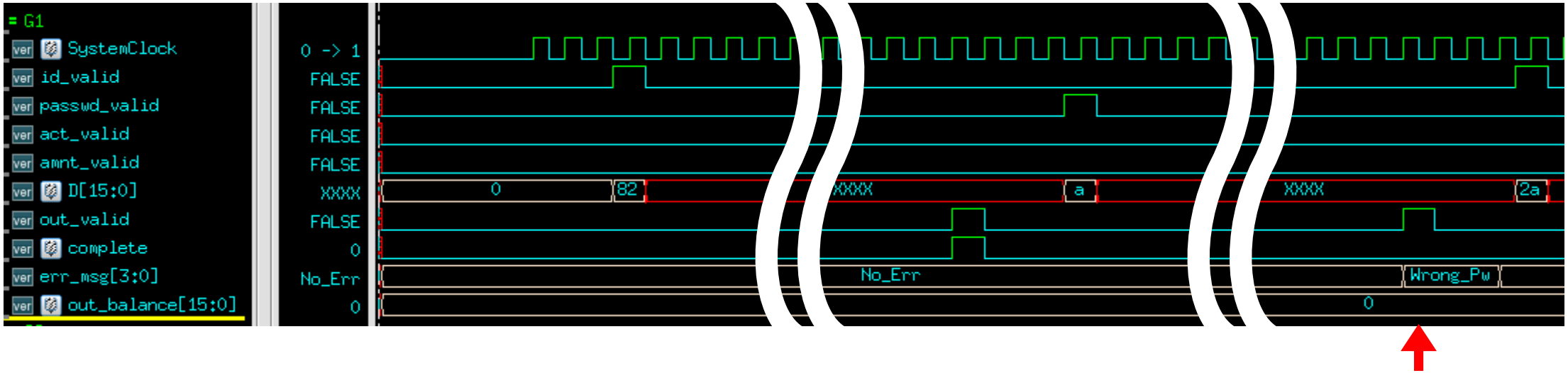
Bank = 4'h 0
Acct_no = 4'h 2
passwd = 4'b 0000
(unregistered)

Bank = 4'h 2
Acct_no = 4'h a
passwd = 4'b 0010
Balance = 16'h 2a2a

Case 1 – user log in, but acct not exist

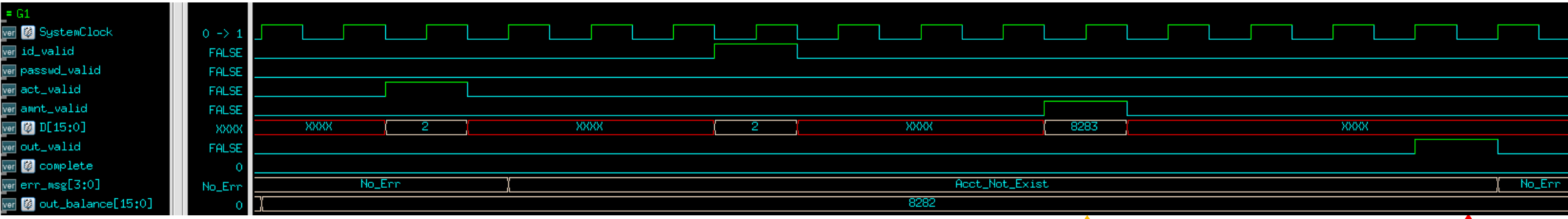


Case 2 – user log in, but password is wrong



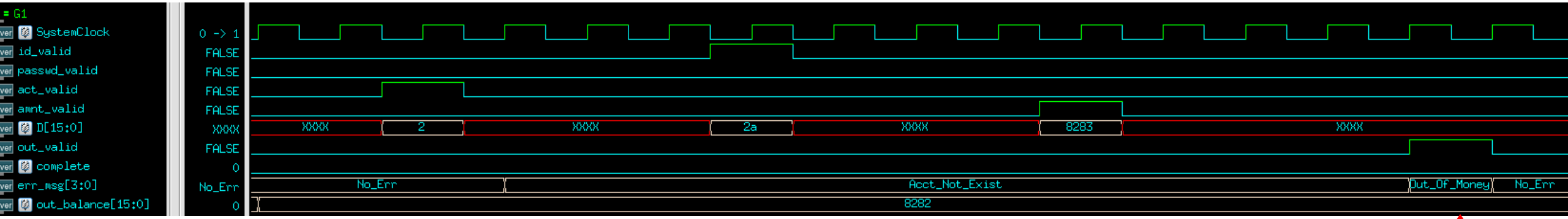
※ If user log in with wrong password, next operation starts from enter ID.

Case 3 – transfer money to unregistered account

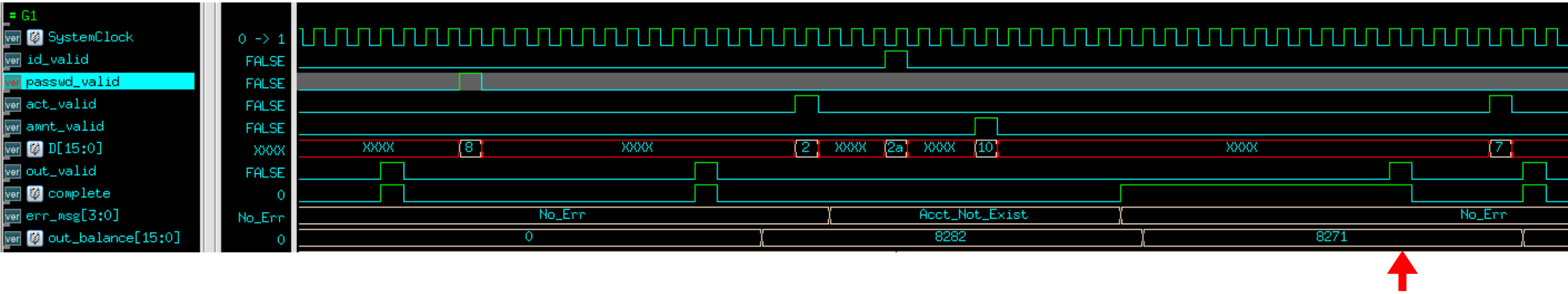


If both acct and amount are illegal,
we only show “acct not exit”

Case 4 – transfer too much money

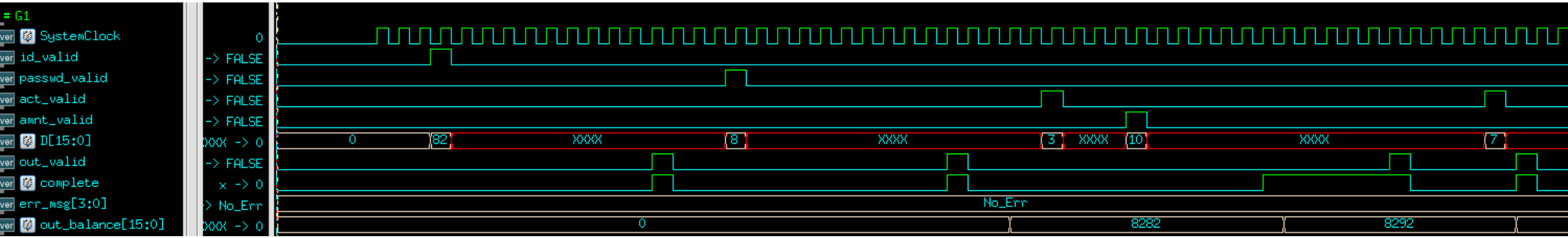


Case 5 – transfer money to acct under different bank

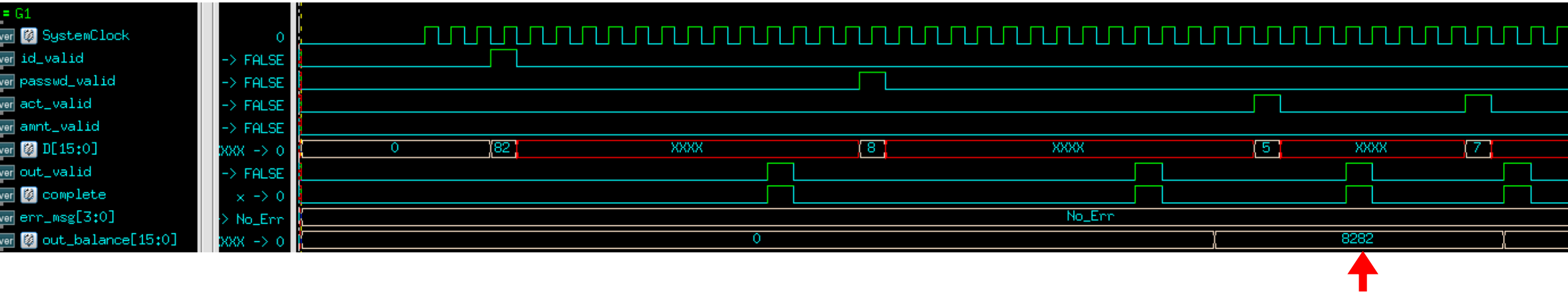


1. If action is transfer money, id_valid will be sent before amnt_valid, and out_valid can’t rise before amnt_valid is given.
2. The gap length between id_valid and amnt_valid is not fixed, at least 1 cycle.

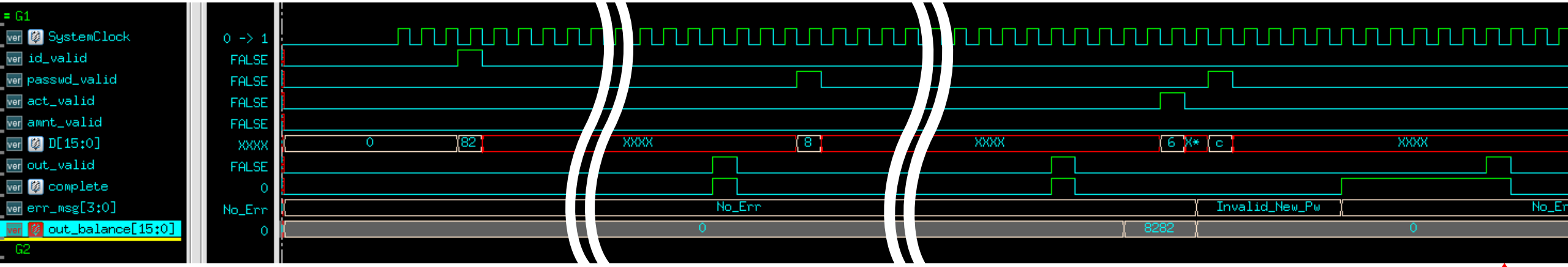
Case 6 – make a deposit



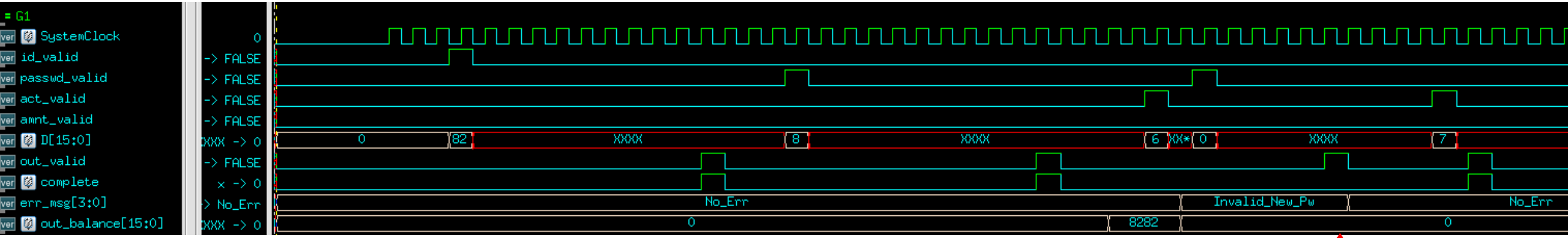
Case 7 – check balance



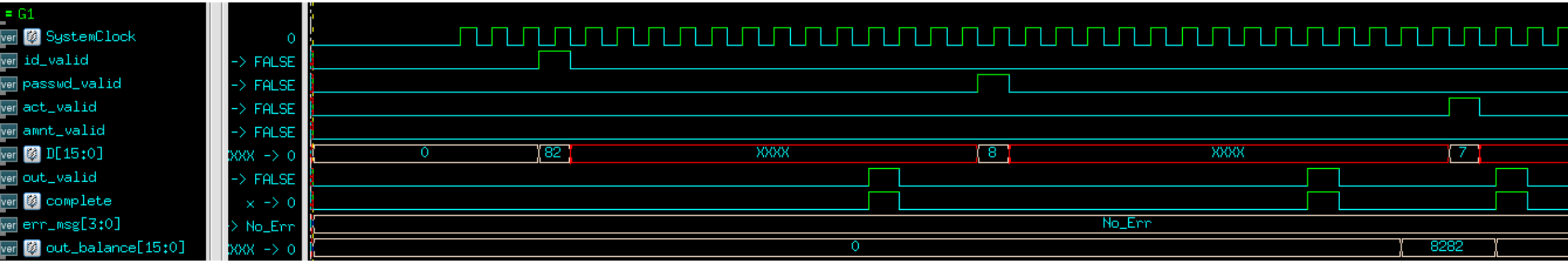
Case 8 – change password successfully



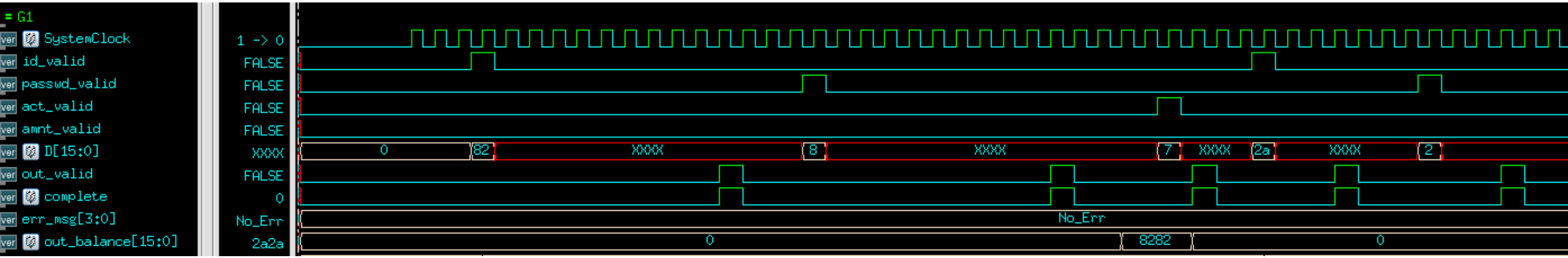
Case 9 – change password, but new password is illegal



Case 10 – log in then directly exit



Case 11 – after exit, next user log in



User rule

- Users can not transfer money to themselves.
- Next operation will be valid 3-5 cycle after out_valid fall.
- The input amount won’t make the system overflow.

System rule

- if action complete, err_msg must be 4’b0.
- If action is not
  - show balance
  - transfer money
  - make a deposit

  out _balance should be 0 when out_valid is high.