SEMICON
THINKING OUT OF THE BOX

# Timer Design Training Project

**Semicon Solutions Co., Ltd**

**Dang Tuong Duong**

# Functions

- interrupt tmr_ovf is active when counter is overflow
- interrupt tmr_udf is active when counter is underflow
- 8-bit Timer can count with 16 internal clock
- 8-bit Timer will be updated when the data register is updated

# Inputs/Outputs (1/3)

- System inputs
  - Inputs:
    - sys_clk
    - rst_n
- Internal clock inputs
  - Inputs
    - clk_0
    - clk_1
    - ……
    - clk15

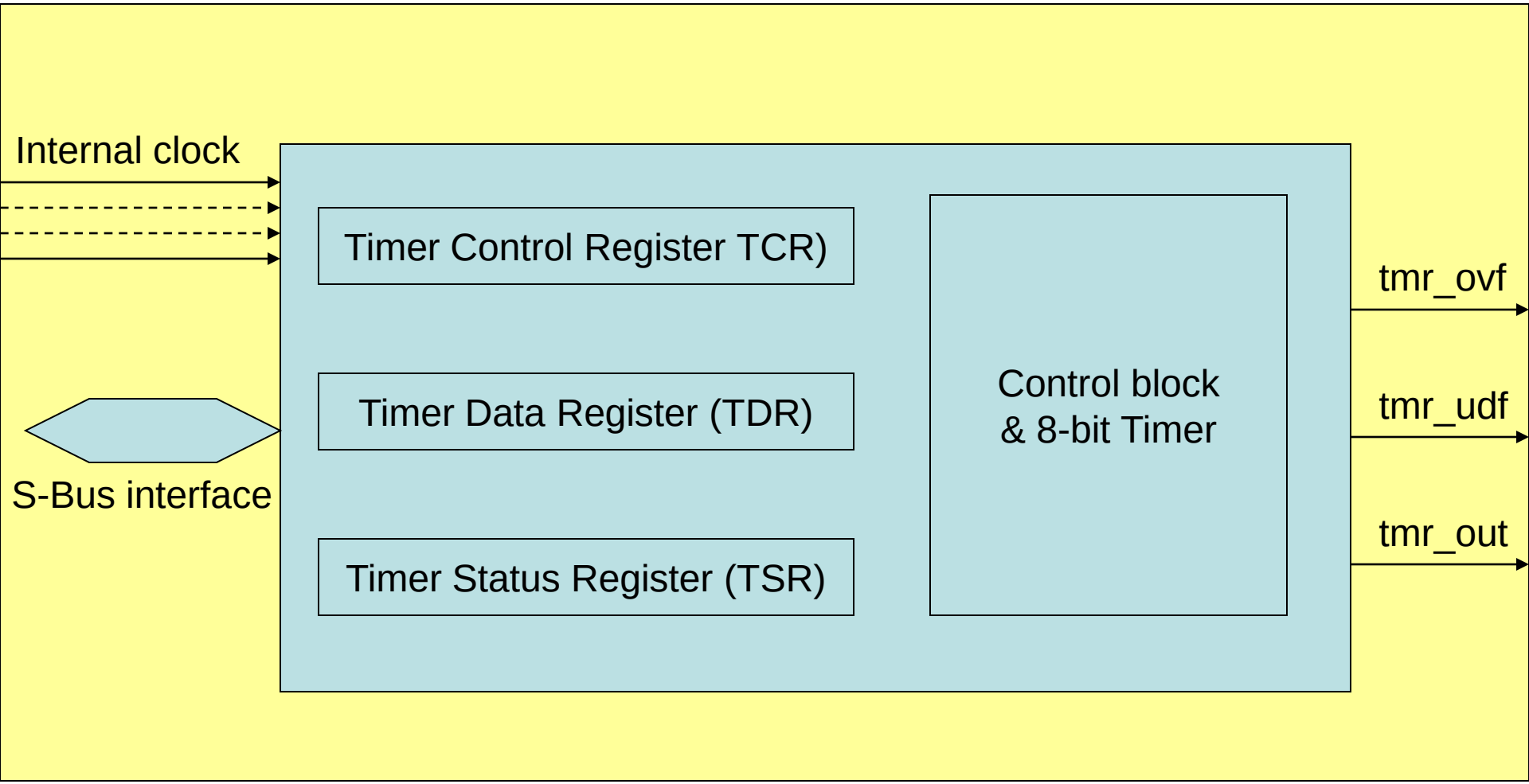
# Inputs/Outputs (2/3)

- S-bus interface signals
  - Inputs
    - chip_select_n
    - read_n
    - write_n
    - size_n
    - address[1:0]
    - wdata[15:0]
  - Outputs
    - rdata[15:0]

# Inputs/Outputs (3/3)

- Module Port
  - Outputs
    - tmr_ovf : overflow interrupt
    - tmr_udf: underflow interrupt
    - tmr_out: output pulse

# Block diagram



# Register Summary

- Timer Control Register (TCR) 2’b00
- Timer Data Register (TDR) 2’b01
- Timer Status Register (TSR) 2’b10
- Reserved 2’b11

# Registers (1/3)

- Timer Control Register (TCR): 8-bit register
  - Bit [3:0]: selects 16 internal clocks
    - 4’b0000: select clk_0
    - 4’b0001: select clk_1
    - ……..
    - 4’b1111: select clk_15
  - Bit [4:6]: enable port: tmr_ovf, tmr_udf, tmr_out
  - Bit [7] :
    - 1’b1: disable counting
    - 1’b0: enable counting
  - Initial value of this register is 8’h00

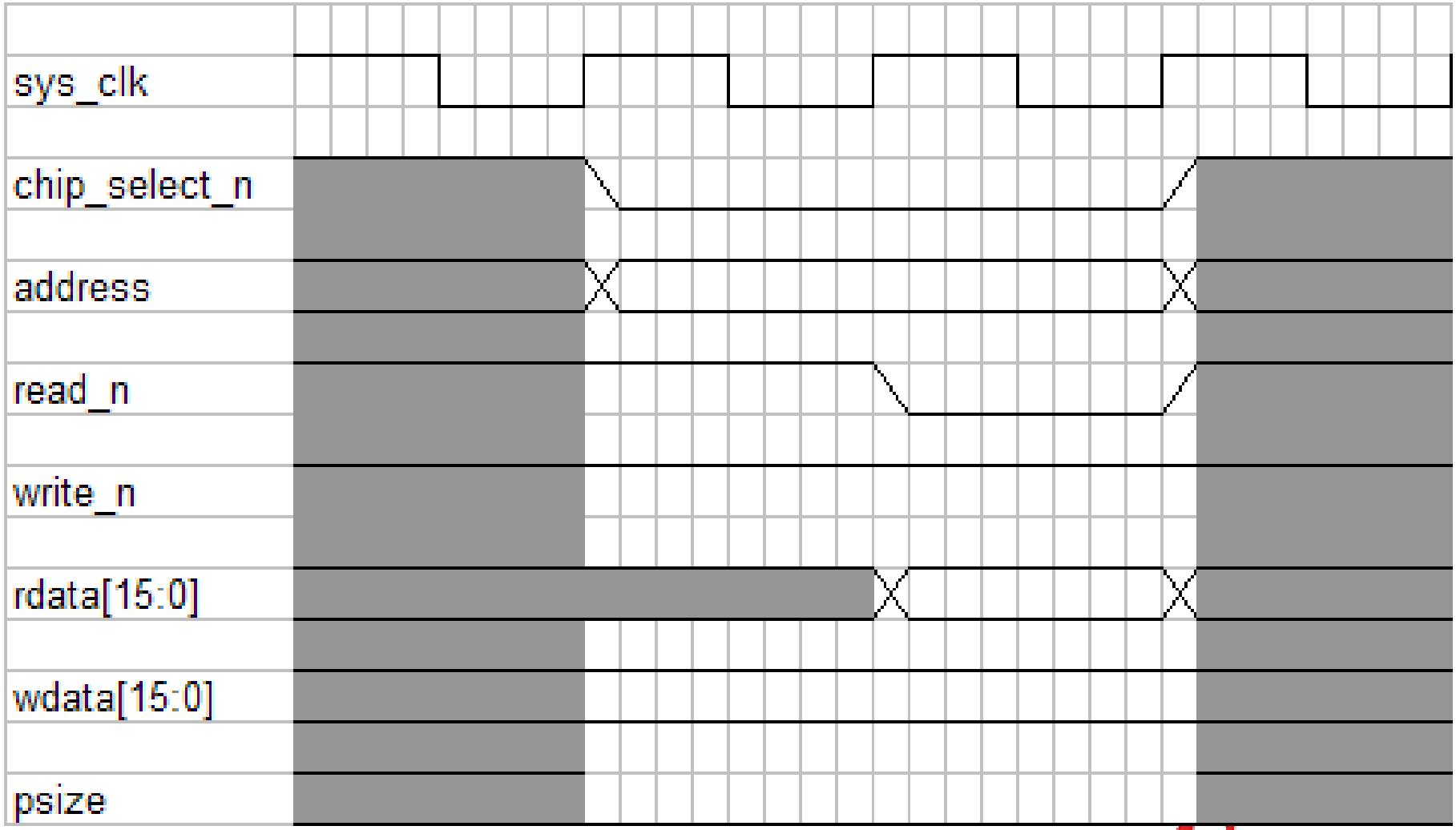
# Registers (2/3)

- Timer Data Register (TDR): 8-bit register
  - Bit [7:0]: Data value is updated to 8-bit counter. If the value of this register is changed, its new value is updated to 8-bit timer immediately
  - Initial value of this register is 8'h00

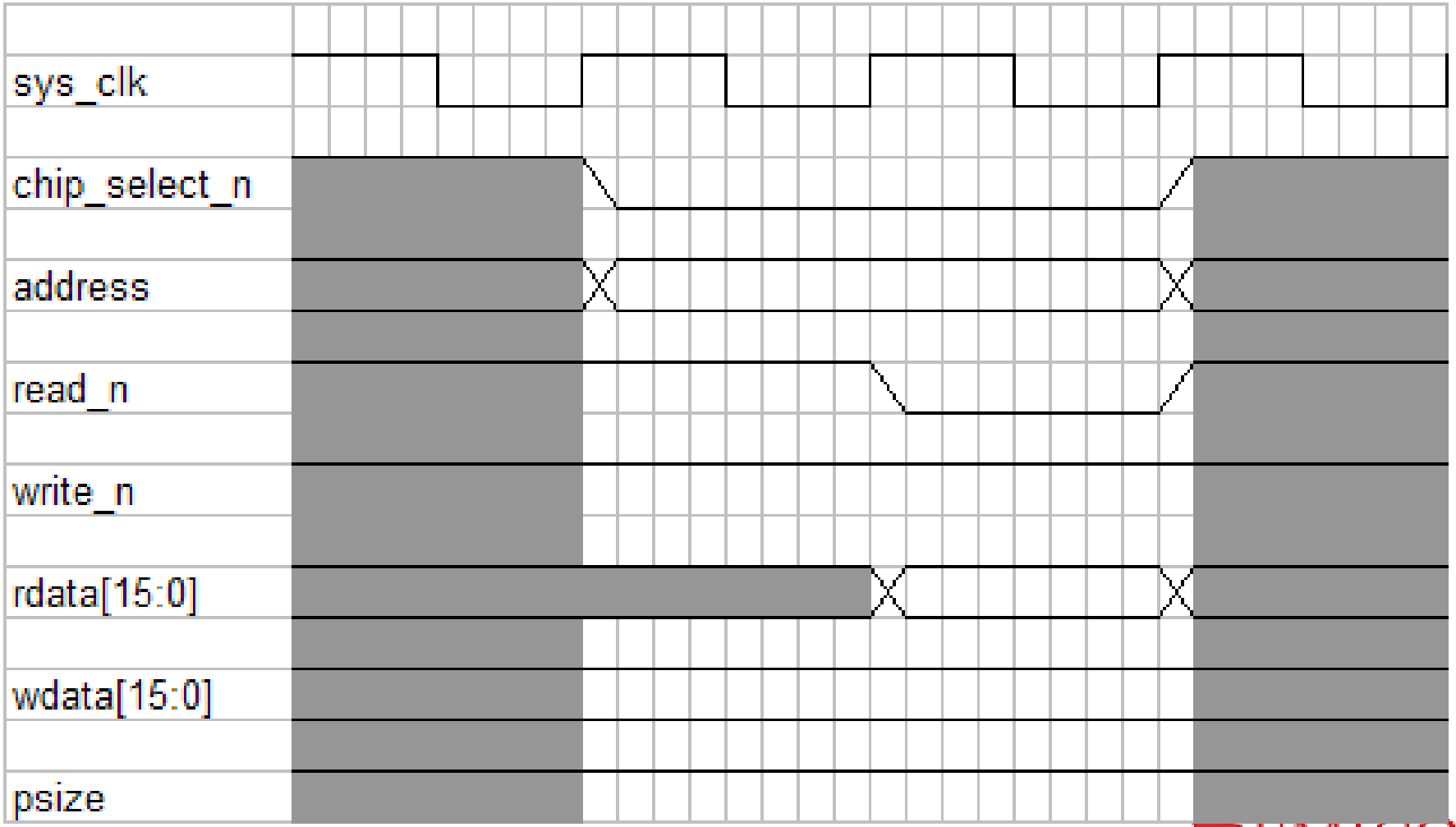
# Register (3/3)

- Timer Status Register(TSR): 8-bit register
  - Bit[7]: status of tmr_ovf
    - Setting condition: 8-bit counter is overflow
    - Clearing condition: Write 1’b0 when this bit is 1’b1 only
  - Bit[6]: status of tmr_udf
    - Setting condition: 8-bit counter is underflow
    - Clearing condition: Write 1’b0 when this bit is 1’b1 only
  - Bit [5:0] : are reserved bits
  - Initial value of this register is 8’h00

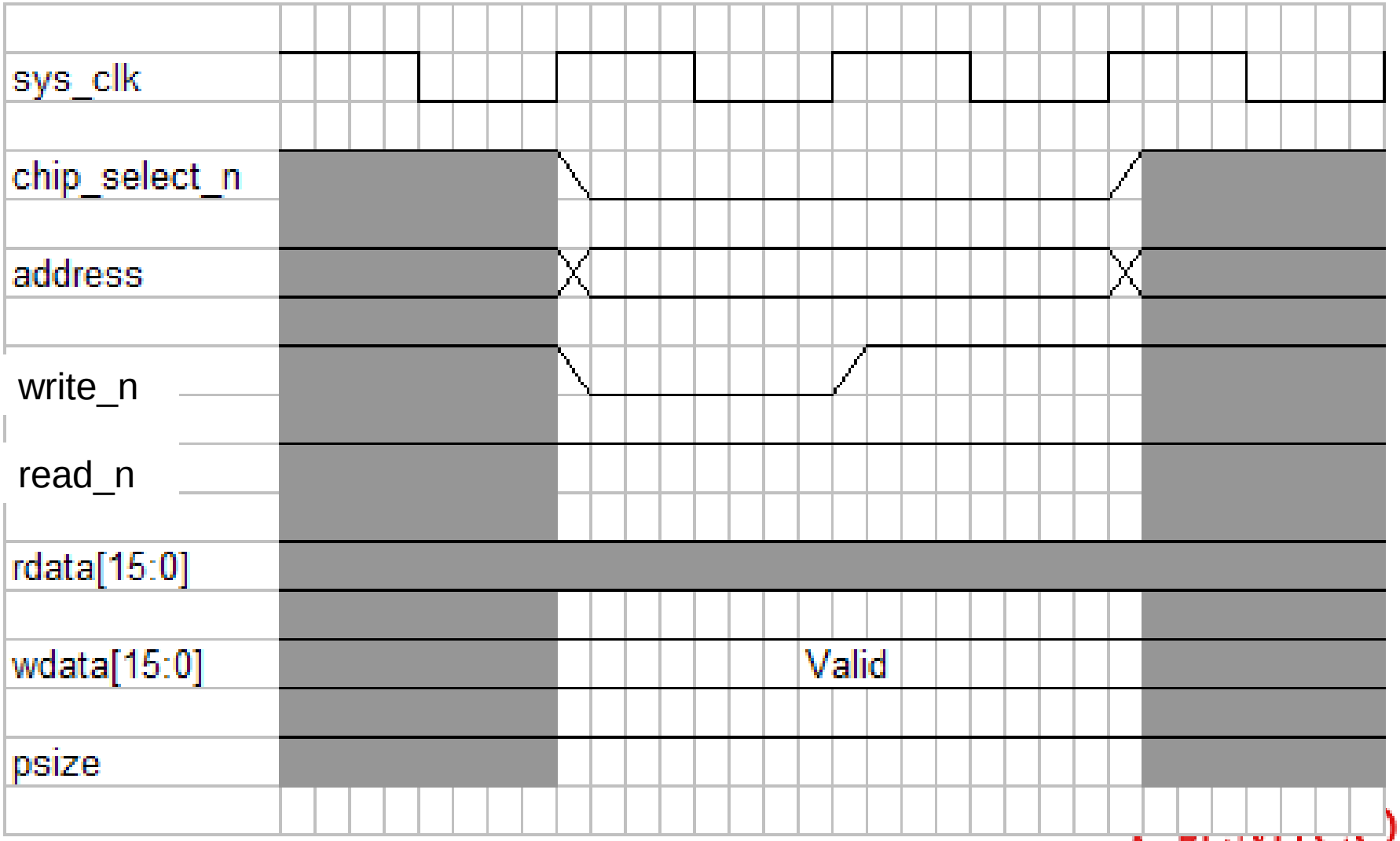
# S-Bus protocol
# (Read cycle 8-bit)



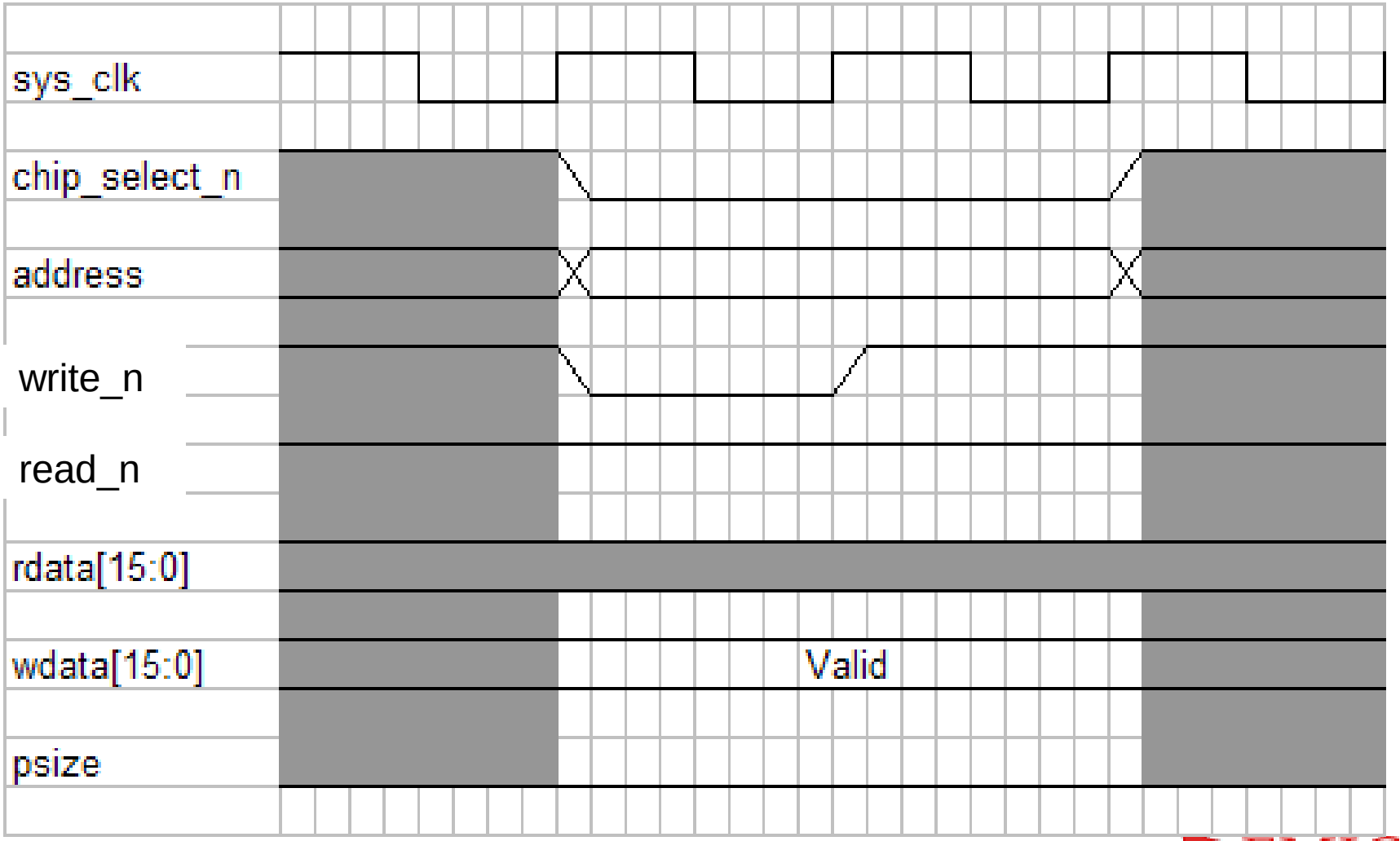
# S-Bus protocol
# (Read cycle 16-bit)



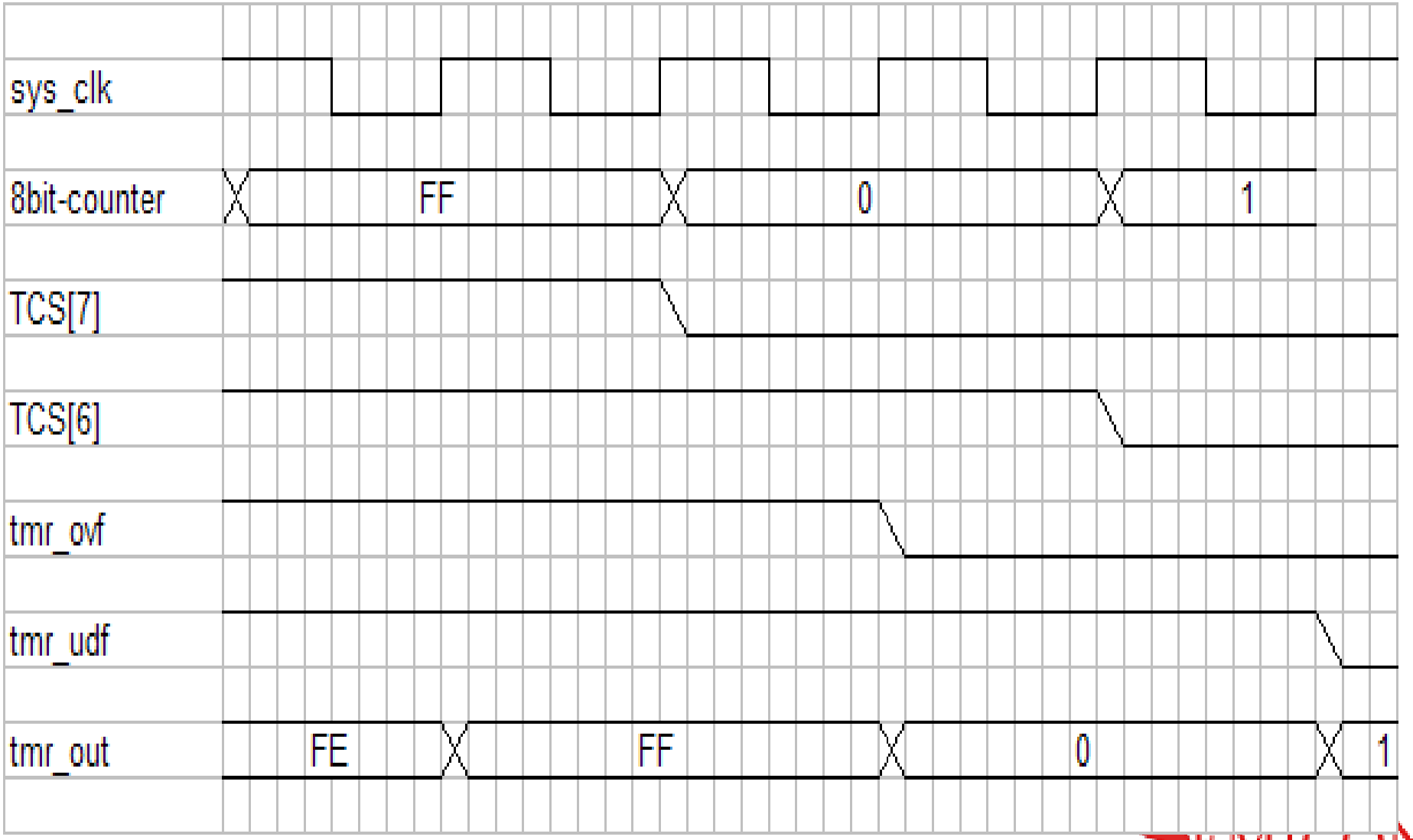
# S-Bus protocol
# (Write cycle 8-bit)



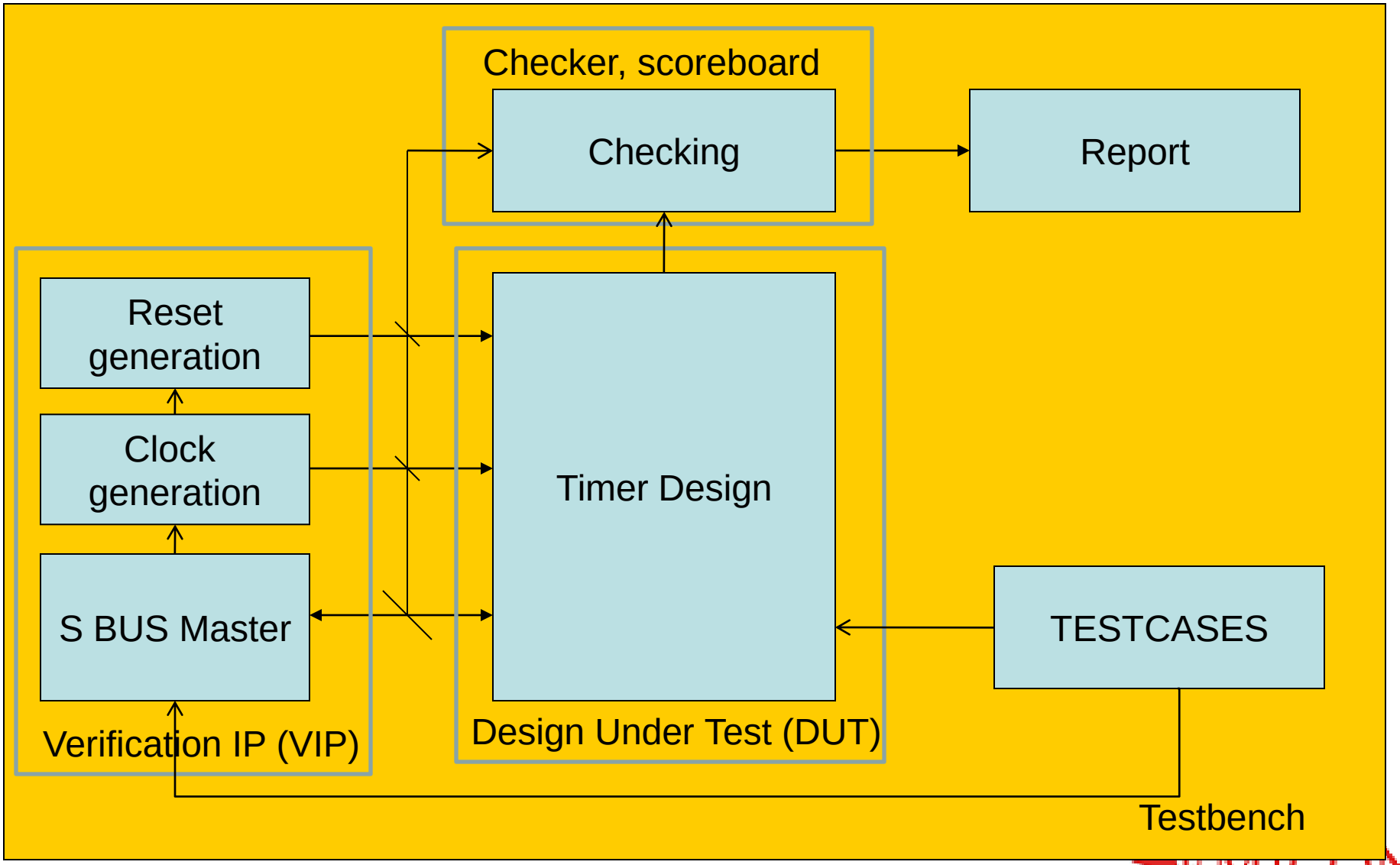
# S-Bus protocol
# (Read cycle 16-bit)



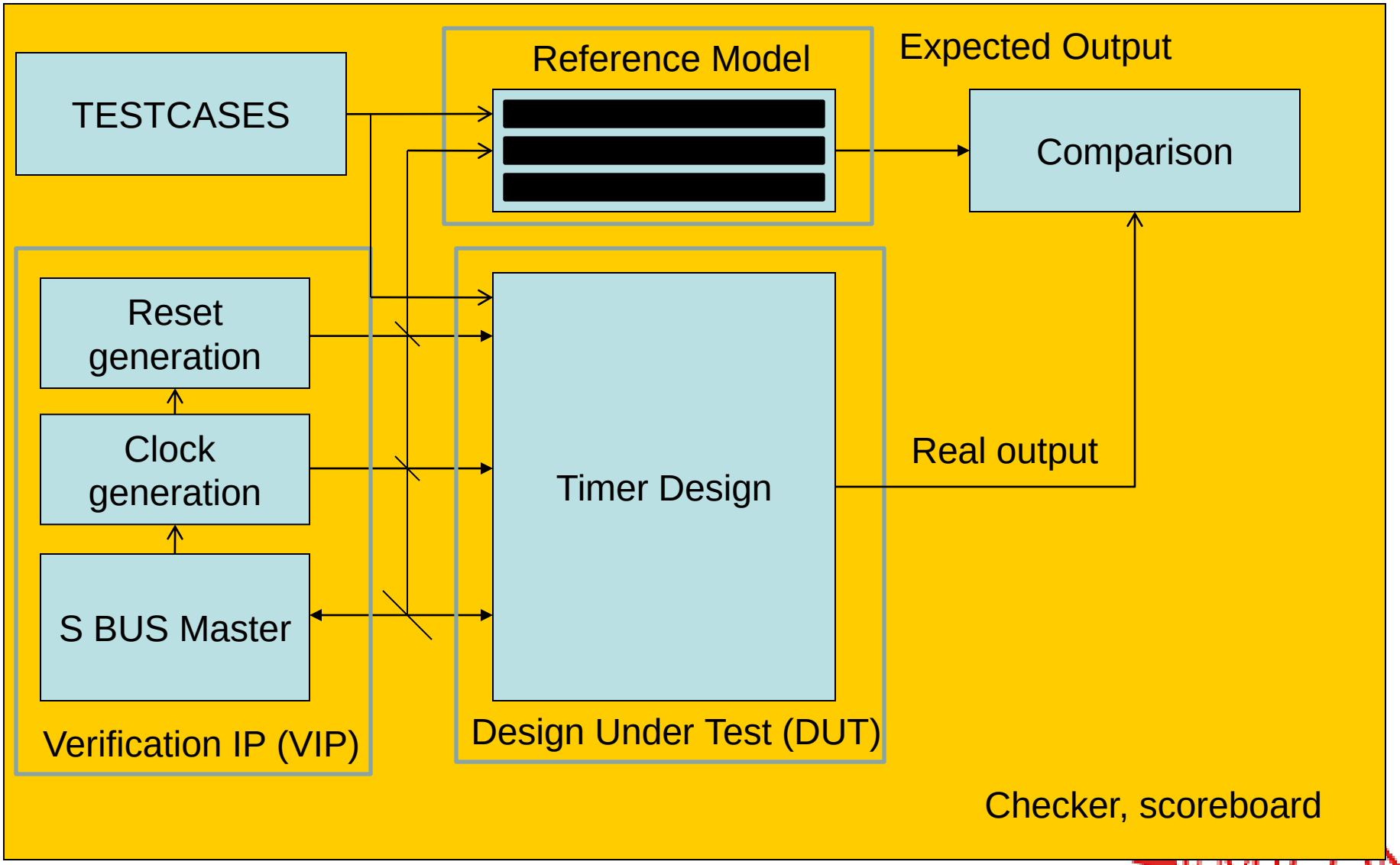
# Operations


sys_clk
8bit-counter
FF
0
1
TCS[7]
TCS[6]
tmr_ovf
tmr_udf
tmr_out
FE
FF
0
1

# Simulation Environment



# Scoreboard Idea



# Structure of SE

- tmr_design
  - rtl (verilog files, VHDL files): RTL directory
  - testbench(top.v, checker, scoreboard): testbech files
  - testcases(*.vt): testcases files
  - sim (script files, log file, executing file): simulation execution
  - vip(verification IP): necessary verification ip

# How to execute tools (Modelsim/Questasim of Mentor)

- Step 1: Check work directory, if not existed, building work dir (use command **vlib**)
- Step 2: Compile RTL files and testbench files (use command **vlog**)
- Step 3: Run simulation (use command **vsim**)

# Command DOS in Windows (run.bat)

**## delete trash files and dirs**

del *

**## Create LIB**

vlib work

**## Mapping WORK dir**

vmap work work

**## Compile RTL and testbench file list**

vlog –f list_rtl.file  ## RTL files

vlog –f list_tb.file  ## testbench files

**## Run simulation with testbench top module name is CPU_TOP**

vsim -l  **test**.log –c CPU_TOP -voptargs=+acc -novopt -assertdebug -do "log -r -d 6 /*; run –all”

Remember to name this file with TAG .BAT

Terminal: **./run.bat**

# run_all.bat

copy test_1.vt run_test.vt

run.bat

ren test.log test_1.log

ren vsim.wlf test_1.wlf

del run_test.vt

copy test_2.vt run_test.vt

run.bat

……

# Cshell Scripts Language (run.csh)

```
#!/bin/csh –f
## delete trash files and dirs
rm –rf *
## Create LIB
vlib work
## Mapping WORK dir
vmap work work
## Compile RTL and testbench file list
vlog –f list_rtl.file  ## RTL files
vlog –f list_tb.file  ## testbench files
## Run simulation with testbench top module name is CPU_TOP
vsim -l  $1.log –c CPU_TOP -voptargs=+acc -novopt -assertdebug -do "log -r -d
    6 /*; run –all”
```

After completing this file, using command **chmod +x run.csh** so that moving it to executing file

Terminal: **./run.csh test_register** → testregister = **$1**

# Bash shell? makefile? How to use?

- Find label: normal

normal: **clean $(CMP_CMD)**

vsim ${DENALI_OPTION} -l ${TEST_NAME}.log -c TB_tp -voptargs=+acc -novopt -assertdebug -do "log -r -d 6 /*; run -all"

echo 'Test Name: $(TEST_NAME)' >> $(TEST_NAME).log

→ Jump to label: **clean** (finish all command in this label)

-> Return label: normal and jump to second label: **$(CMP_CMD)**

**-> After finish all command in second label, jump back label normal, and execute command of label normal**

1- vsim ${DENALI_OPTION} -l ${TEST_NAME}.log -c TB_tp -voptargs=+acc -novopt -assertdebug -do "log -r -d 6 /*; run -all"

echo 'Test Name: $(TEST_NAME)' >> $(TEST_NAME).log

# Flow of sim/Makefile

**LABEL_1 = compile #→ assign variable**
**LABEL_2 = compile.f #→ assign variable (list all files need to be compiled)**

```
normal: clean $(LABEL_1)
        vsim -l ${TEST_NAME}.log -c TB_tp -voptargs=+acc -novopt -assertdebug -do "log -r -d 6 /*; run -all"
        echo 'Test Name: $(TEST_NAME)' >> $(TEST_NAME).log

compile: create_lib
        vlog +incdir+$(INC_DIR) -f $(LABEL_2)

create_lib:
        test -e work || vlib work
clean:
        rm -f run_test.vt run_test.def
        cp -f ../testcases/$(TEST_NAME).vt  run_test.vt
        rm -rf  *.tmp  transcript work *.wlf vsim.fcdb
regress:
        sed '/^#/d' $(TEST_LIST) > ./tmp.lst
        make regress_core TEST_LIST=tmp.lst
regress_core:
        for i in ${REGRESSION_LIST}; do \
       echo $$i ; \
       make "TEST_NAME=$$i"; \
       done
         make report TEST_LIST=tmp.lst
         rm -f ./tmp.lst
report:
         rm -f regress_report.list
         for i in ${REGRESSION_LIST}; do \
       echo -n $$i " : "  >> regress_report.list; \
       sed -n -e '/TEST PASSED/p' $$i.log >> regress_report.list ;\
       sed -n -e '/TEST FAILED/p' $$i.log >> regress_report.list ;\
       done
         cat regress_report.list
```

# compile.f

- tmr_design
  - rtl (*.v): RTL directory
  - testbech(top.v): testbech files
  - testcases(*.vt): testcases files
  - sim: simulation execution
  - vip(master.v, clock.v): verification ip

Current dir is sim

../rtl/a.v

../rtl/b.v

../testbench/*.v

../vip/*.v

# Flow of environment

- Do task gen_rst() in top.v
- Do task run_task() in testcase_name.v
- Do task report() in top.v
- Stop simulation

# top.v – Procedure of simulation

```
……..
// sumilation procedure
initial begin
      test_status = 32'hABABABAB;
// Init
      #10;
      gen_rst;
// run testcases
      run_test;

      #100;
// report
      report_test(1);
   end
…….
……….
```

# top.v - Structure

```
module top();
Integer count = 0;
`include “run_test.vt” // include testcase

// sumilation procedure
initial begin
      test_status = 32'hABABABAB;
      #10;
      gen_rst;
      run_test;
      #100;
      report_test(1);
   end

task report_test;
  Input enable;
  If (count =! 0) report_fail();
  else report_pass();
endtask

task report_pass;
$display(“TEST PASSED”); $finish()
endtask

task report_fail;
$display(“TEST FAILED”); $finish()
endtask
endmodule
```

# Some tasks in master.v

- write(address, data, size)
  - Address: 2 bits address
  - Data: 16 bits write data
  - Size: size of data bus
    - 0: 8 bits data bus
    - 1: 16 bits data bus

# Some tasks in master.v

- read(address, data, size)
  - Address: 2 bits address
  - Data: 16 bits read data
  - Size: size of data bus
    - 0: 8 bits data bus
    - 1: 16 bits data bus

# Format of testcase

```
task run_test;
    reg [7:0] data;

        Top.cpu_model.MOVT(2'b00, 8'h44);
        Top.cpu_model.MOVF(2'b00,data);
        If (data =! 8'h44) count = count + 1;
        Top.cpu_model.MOVT(2'b01, 8'hBB);
        Top.cpu_model.MOVF(2'b01,data);
        If (data =! 8'hBB) count = count + 1;
        MOVT(2'b02, 8'hAA);
        MOVF(2'b02,data);
        If (data =! 8'hAA) count = count + 1;
        ….
endtask
```

# How to run simulation

- make TEST_NAME=<test_name>
- make regress TEST_LIST=<test_list>

# Testcase Plan

- Write data to TCR, read TCR, compare write data and read data (repeat many times)
- Write data to TDR, read TDR, compare write data and read data (repeat many times)
- Write data to TSR, read TSR, compare write data and read data (repeat many times)

# Testcase Plan

- Test operation of timer
  - Register Read/Write Accessing tests
  - Counting up
  - Counting down
  - Disable counting
  - Enable counting
  - Overflow interrupt
  - Underflow Interrupt
  - Status register updating
  - Timing of counter according to each external clock selection

# Coverage

- Run coverage for all test

Thank you for your attention!

SEMICON
THINKING OUT OF THE BOX