

IDT 70T3509M SYNCHRONOUS DUAL-PORT

STATATIC RAM Testbench Structure

This document describes the structure of test environment that will be used to verify the functional behavior and timing of the IDT 70T3509M FIFO memory VITAL model.
Test environment has two main parts: Package, that generates abstract stimuli information and Test bench that interfaces directly to the model under test, driving logical stimuli and accepts its responses.

Document version: 1.0

Author's name: Vladeta Ljubisavljevic

Author's e-mail: v-ljubisavljevic@hdl-dh.com

File location: /cvs/vital/local/idt70t3509m/src

HDL Design House, Belgrade, SCG

Rev	Date	Author	Note
1.0	23/09/2005	Vladeta Ljubisavljevic	Initial version

*This document is confidential property of HDL Design House.
Do not disclose, copy, reproduce or distribute its contents without permission.*

Copyright 2005 by HDL Design House



HDL Design House
Makenzijeve 79
11000 Belgrade,
Serbia and Montenegro,
Tel: +381 (0)11 245-7998
Fax: +381 (0)11 245-9987
e-mail: office@hdl-dh.com

Table of Contents

1	Package-----	3
1.1	Command data type-----	3
1.2	TC Selection-----	5
1.3	Aux type-----	5
1.4	Checker procedures-----	5
2	Testbench-----	6
2.1	Test parameters-----	6
2.2	Testbench drivers-----	6
2.3	Testbench checkers-----	7

1 Package

Package defines several data types and procedure **Generate_TC**. Primary functionality of that procedure is to generate command sequence for defined test cases. Sequence is passed to test bench as a global variable.

1.1 Command data type

```
TYPE cmd_rec IS  
    RECORD  
        cmdA      : cmd_type;  
        cmdB      : cmd_type;  
        statusA   : status_type;  
        statusB   : status_type;  
        beA       : NATURAL;  
        beB       : NATURAL;  
        dataA1    : NATURAL;  
        dataA2    : NATURAL;  
        dataA3    : NATURAL;  
        dataA4    : NATURAL;  
        addressA  : NATURAL;  
        dataB1    : NATURAL;  
        dataB2    : NATURAL;  
        dataB3    : NATURAL;  
        dataB4    : NATURAL;  
        addressB  : NATURAL;  
        aux       : Aux_type;  
        wtime     : time;  
    END RECORD;
```

Each command is performed in sequence. A stands for Left port, B stands for right port. Status field is used for read and write operation. Commands are performed in parallel on left and right port. Because of that we have two commands merged in one record for left and right port. Detailed description of fields follow:

1. cmdA, cmdB
 - command for left or right port
 - . **done** - Indicate that command sequence is over. This is default command.
 - . **idle** - All command signals are deactivated, and address and data are tristate...
 - . **none** - This is no action command. This command is used when we don't want to drive on one port anything, but on other to drive valid data.
 - . **rdPipe** - Read pipeline command. For turning on checking of read data, one would set status field in command on read.

- **rdFT** - Read flow through operation. For turning on checking of read data, one would set status field in command on read.
 - **rdCNTFT** - Read flow through with address increment pin set. First we read from address in address field of command. On next clock cycle address is incremented and data is read from that address.
 - **rdCNTPIPE** - Read pipeline. Same as rdCNTFT except that pipeline pin is set. Everything is delayed for one cycle of clock signal.
 - **rdREPEAT** - Read with address reload. Address is now address stored in internal address register.
 - **setInt** - Set interrupt flag. Write to locations FFFFFh or FFFFEh depending for which side of dual port ram.
 - **resetInt** - Reset interrupt flag. Read from same locations from previous bullet.
 - **sleepMod** - Set ZZ pins to enter sleep mode.
 - **wrt** - Write command. It write data to memory on address specified by command address field. If status is error, data will not be written to testbench memory.
 - **sync_clks** - Command to synchronize clocks. Clocks are after this command separated for 1 ns. Skew between clocks must be 6 ns. So this command is used to make collision situation on write to memory command.
 - **start_clk** - Command for enabling of clock.
 - **stop_clk** - Command for stopping of clock.
 - **change_trigg_clk** - Command for changing triggering clock for test bench. Default triggering clock is left clock. It can be changed with this command.
 - **w_cnten** - Command for enabling or disabling of CNTNeg pins.
 - **w_rep** - Command for enabling or disabling of REPEATNeg pins.
 - **w_zz** - Command for enabling or disabling of sleep pins ZZ.
2. statusA, statusB - status for read and write operation. It can take several values.
- ```
-- list of data/ status to be verified by checker
TYPE status_type IS (
 none, --nothing to check
 readX, -- for reading X values on bus
 error --valid in combination with write
);
```
- **none** - This is default status. Read checker is enabled
  - **readX** - Enable checker for checking X values on data bus.
  - **error** - Disabling of writing to test bench memory.
3. beA, beB - Byte enable field for left and right port. It is used to set which byte is valid for read or write operation.
4. dataA1, dataA2, dataA3, dataA4, dataB1, dataB2, dataB3, dataB4 – Data bus parts for left and right port respective.
5. addressA, addressB - Address fields for write and read operations. A stands for Left port, B for right.
6. aux - Field for enabling or disabling of pins...
7. wtime - This field is used in clock synchronization.
- 8.

## 1.2 TC Selection

Which Testcase will be generated by generate\_TC procedure, is decided in Pick\_TC procedure. Testcases are selected sequentially from TS1 to TS8. Information about the number of testcases in each testseries is stored in constant array **tc**.

```
--number of testcases pre testseriee
 TYPE tc_list IS ARRAY (1 TO TS_MAX) OF NATURAL RANGE 0 TO TC_MAX;

 CONSTANT tc : tc_list :=
-- 1-2-3-4-5-6-7-8
 (3,5,3,4,2,2,4,1);
```

## 1.3 Aux type

```
TYPE Aux_type IS (valid,
 disable);
```

**AUX** parameter is used to transfer some additional information, about current instruction, to test bench.

**Valid** is the default value.

**Disable** value is used:

- with w\_CE command to set or disable CEx pins
- with w\_zz command to set or disable ZZx pins
- with w\_rpt command to set or disable REPEATxNeg pins

## 1.4 Checker procedures

Package checker procedures that are called by testbench to verify model's responses. Those procedures accept model's outputs (IOx, BEx).

Procedures are defined as follows:

```
PROCEDURE Check_read (
 IO : IN std_logic_vector(35 downto 0);
 Data0 : IN NATURAL;
 Data1 : IN NATURAL;
 Data2 : IN NATURAL;
 Data3 : IN NATURAL;
 BE : IN NATURAL;
 SIGNAL check_err : OUT std_logic);

PROCEDURE Check_X (
 IO : IN STD_LOGIC_VECTOR(35 DOWNT0 0);
 BE : IN NATURAL;
 SIGNAL check_err : OUT STD_LOGIC);
```

If checker procedures verify expected output "READ OK" type of message is printed, and if error is detected appropriate error message is generated and check\_err signals is activated. Active value is '1'. This signal goes back to inactive value.

Also, it is possible to check if X values are on output – violation during write operation. When reading, since there are large number of location to be read and verified, there is no need to write message for every read location. write\_msg parameter is used for that purposes.

## 2 Testbench

Testbench interprets abstract data stimuli provided by the generate\_TC procedure and transforms it to actual bus cycles.

### 2.1 Test parameters

Test environment can be used for verification of number of IDT DP models of the 70T3509M family. There are several parameters for configuring Test bench to a particular model.

```

-- DPRam memory configuration

CONSTANT PartID : STRING := "idt70t3509m";
CONSTANT MaxData : NATURAL := 16#1FF#;
CONSTANT MemSize : NATURAL := 1048575;

-- Model configuration

CONSTANT mem_file_name : STRING := "idt70t3509m.mem";
CONSTANT UserPreload : BOOLEAN := TRUE;
CONSTANT TimingModel : STRING := "idt70t3509ms133bpi";
CONSTANT tRead : TIME := 15 NS;
```

**MaxData** parameter holds the maximum value to be stored in a memory cell.

**MemSize** parameter holds the number of location in memory.

**tRead** this parameter holds time to wait after clock rise event to read data on data bus.

### 2.2 Testbench drivers

Every command of the command sequence generated by **generate\_TC** procedure is passed to the **cmd\_dc** procedure.

**cmd\_dc** procedure breaks each command into one or more model specific bus cycles. Actual bus interfacing is done through **bus\_cycle** procedure that drives one of five defined bus cycles. Command is passed to this procedure from the command sequence or are generated in the **cmd\_dc** procedure (for model's command cycles)

```
TYPE bus_type IS
 (bus_idle,
 bus_lidle,
 bus_ridle,
 bus_lwrite,
 bus_rwrite,
 bus_lpread, -- Pipeline read operation left port
 bus_rpread, -- Pipeline read operation right port
```

```
bus_lfread, -- Flow-Through read operation left port
bus_rfread, -- Flow-Through read operation right
 -- port
bus_addrLatchR, -- Address latching right port
bus_addrLatchL, -- Address latching left PORT
bus_addrIncl, -- Increment address left PORT
bus_addrIncr, -- Increment address right PORT
bus_repeatL, -- Repeat address left PORT
bus_repeatR -- Repeat address right port
);
```

Defined bus cycles are:

**bus\_idle** Bus idle cycle. All control signals are driven to their inactive values.  
**bus\_lidle** Bus idle cycle. All control signals for left port are driven to their inactive values.  
**bus\_ridle** Bus idle cycle. All control signals for right port are driven to their inactive values.  
**bus\_lfread** Bus flow through read cycle for left port. Set control signals on active level if valid command, else on inactive level.  
**bus\_rfread** Bus flow through read cycle for right port. Set control signals on active level if valid command, else on inactive level.  
**bus\_lpread** Bus pipeline read cycles for left port. Set control signals on active level if valid command, else on inactive level.  
**bus\_rpread** Bus pipeline read cycles for right port. Set control signals on active level if valid command, else on inactive level.  
**bus\_lwrite** Bus write cycle. Set control signals on active level if valid command, else on inactive level.  
**bus\_rwrite** Bus write cycle. Set control signals on active level if valid command, else on inactive level.  
**bus\_addrLatchR** Address latching on right port.  
**bus\_addrLatchL** Address latching on left port.  
**bus\_addrIncr** Address increment on right port. It is used to assert CNTENRNeg port.  
**bus\_addrIncl** Address increment on left port. It is used to assert CNTENLNeg port.  
**bus\_repeatR** Reload address on right port. It is used to enable or disable REPEATNeg pin.  
**bus\_repeatL** Reload address on left port. It is used to enable or disable REPEATNeg pin.

## 2.3 Testbench checkers

In order to verify that correct data is present on the Q lines, and that flags are valid, user have to specify what is expected to be read by that command.

According to that parameter checker process compares Q lines value with data from the memories

If checker gets expected output "READ OK" type of message is printed, and if error is detected appropriate error message is generated and check\_err signals is activated. Active value is '1'. This signal goes back to inactive value.

**Check\_read** procedure checks if valid data is read from memory.

**Check\_X** procedure checks if X are read from data bus.