A Universal DFT Verification Environment: Filling the Gap between Function Simulation and ATE Test

Rui Huang
Advanced Micro Devices
The Background

• ATEs need test patterns described by STIL (Standard Test Interface Language) or other test languages.

• In DFT domain, the test patterns running on an ATE can be categorized in two types: scan related and none-scan related.

• Scan related test patterns can be generated using ATPG tools.
The Issue with Non-scan Related Test Patterns…

• Non-scan related test patterns are normally created by design verification engineers using languages such as System Verilog or C++.

• How this is done in practice?
The Issue with Non-scan Related Test Patterns…

• A dedicated team to transfer function simulation to ATE test environment.
• Some in-house automation flows are developed to enforce complex rules on test writing and register specification documentation.

• Is there a more efficient and universal solution?
A More Efficient and Universal Solution

dft_test

- jtag_agent
  - jtag_seqr
  - jtag_mon
  - jtag_drv

- pad_agent
  - pad_seqr
  - pad_mon
  - pad_drv

- rst_agent
  - rst_seqr
  - rst_mon
  - rst_drv

configuration

Environment

DUT

clk_drv

pad_agent

jtag_agent

sequence
A More Efficient and Universal Solution

configuration

sequence
dft_test

1) JTAG port
2) reset pads
3) clock pads
4) other pads

- jtag_agent
  - jtag_mon
  - jtag_drv
  - jtag_seqr
- pad_agent
  - pad_mon
  - pad_drv
  - pad_seqr
- rst_agent
  - rst_mon
  - rst_drv
  - rst_seqr
- clk_drv
A More Efficient and Universal Solution

**Diagram Description:**

- **dft_test**
  - **jtag_agent**
    - **jtag_seqr**
  - **pad_agent**
    - **pad_seqr**
  - **rst_agent**
    - **rst_seqr**

- **Configuration**
  - **STIL vector**
  - **stil_info_transaction**

- **STIL Pattern**

- **Environment**

- **Ports:**
  - 1) **JTAG port**
  - 2) **reset pads**
  - 3) **clock pads**
  - 4) **other pads**

**Legend:**

- **STIL**
- **generator**
- **sequence**
JTAG FSM is in shift DR state. It will send three bits 101 on TDI and sample TDO with golden compare data (three bits 110).

```
1: @TCK negedge a, TMS = 0; TDI = 1; TDO = X;
2: @TCK posedge b, TMS = 0; TDI = 1; TDO = H;
3: @TCK negedge c, TMS = 0; TDI = 0; TDO = H;
4: @TCK posedge d, TMS = 0; TDI = 0; TDO = H;
5: @TCK negedge e, TMS = 0; TDI = 1; TDO = H;
6: @TCK posedge f, TMS = 0; TDI = 1; TDO = L;
```
An Example of Clock Pads Connection in Testbench

The only exception: clock_driver only drives TCK, other reference clocks are connected from the testbench.

1: TCK = 0; PLL1_REF = 0; PLL2_REF = 0; BYPASS_CLK = 0;
2: TCK = 1; PLL1_REF = 1; PLL2_REF = 1; BYPASS_CLK = 1;
3: TCK = 0; PLL1_REF = 0; PLL2_REF = 0; BYPASS_CLK = 0;
4: TCK = 1; PLL1_REF = 1; PLL2_REF = 1; BYPASS_CLK = 1;
One driver AP’s Ping and Pong buffer are full

Query all driver APs’ keys

Concatenate all Ping buffer strings and update ping/pong_data_rdy and ping/pong buffer.

Put back all keys.

Write concatenated string to STIL pattern file.
DFT TDR (Test Data Register) Abstraction
DFT TDR (Test Data Register) Abstraction
DFT TDR (Test Data Register) Abstraction

LEVEL0 SIB

IEEE 1149 DR

LEVEL1 SIB

IEEE 1500 Client 0

WSI WSI

LEVEL0 SIB1

IEEE 1500 Client 1

WSI WSO

LEVEL1 SIB0

IEEE 1500 Client 2

WSI WSO

LEVEL1 SIB1

IEEE 1500 Client 3

WSI WSO

TDI TDO
DFT TDR (Test Data Register) Abstraction

IEEE 1149 IR
IEEE 1149 DR
LEVEL0 SIB
LEVEL0 SIB1
WSI IEEE 1500 Client 1 WSO
WIR D D D
WDRx D D
SEL_WIR D
OPCODE
LEVEL1 SIB1
WSI IEEE 1500 Client 3 WSO
WIR D D D
WDRx D D
SEL_WIR D
LEVEL1 SIB0
WSI IEEE 1500 Client 2 WSO
WIR D D D
WDRx D D
SEL_WIR D
LEVEL1 SIB1
WSI IEEE 1500 Client 0 WSO
WIR D D D
WDRx D D
SEL_WIR D
LEVEL0 SIB
WIR D D D
WDRx D D
SEL_WIR D
OPCODE
1
TDI TDO
DFT TDR (Test Data Register) Abstraction
DFT TDR (Test Data Register) Abstraction

IEEE 1149 DR

IEEE 1500 Client 0

IEEE 1500 Client 1

IEEE 1500 Client 2

IEEE 1500 Client 3

LEVEL0 SIB

LEVEL1 SIB

LEVEL0 SIB0

LEVEL0 SIB1

LEVEL1 SIB0

LEVEL1 SIB1

WDRx

SEL_WIR

WIR

D

D
DFT TDR (Test Data Register) Abstraction
DFT TDR (Test Data Register) Abstraction
DFT TDR (Test Data Register) Abstraction
DFT TDR Layer

TDR address encode example

TDR OPCODE
LEVEL1_SIB1
LEVEL1_SIB0
LEVEL0_SIB1
LEVEL0_SIB0

SIB code

dft_tdr_predictor

dft_tdr_transaction

dft_tdr_monitor

dft_tdr_network

jtag_transaction

dft_tdr_adapter

dft_tdr_sequencer

dft_tdr_transaction

up_sequencer

dft_tdr_trans_to_jtag_trans_sequence

jtag_transaction

3/2/2022

Rui Huang, Advanced Micro Devices
The bit value stores update register value and the bit out stores shift register value.

```vhdl
class sib_node extends uvm_object;
    `uvm_object_utils(sib_node)
    bit in0;
    bit in1;
    bit value;
    bit out;

    function new(string name = "sib_node");
        super.new(name);
    endfunction : new

    function void out_update ();
        out = value ? in1 : in0;
    endfunction: out_update

    function void value_update ();
        value = out;
    endfunction: value_update

endclass : sib_node
```

The `out_update` function simulates the active clock edge on shift register and the `value_update` function simulates the active clock edge on update register.
class reg_node extends uvm_object;
  `uvm_object_utils(reg_node)
  bit in;
  bit is_selwir;
  bit value;
  bit out;
  function new(string name = "reg_node");
    super.new(name);
  endfunction : new
  function void out_update ();
    out = in;
  endfunction: out_update
  function void value_update ();
    value = out;
  endfunction: value_update
endclass : reg_node
Summary

• This UVM-based DFT environment works well in an experiment project and the generated STIL test pattern files pass simulation using STIL Verify.

• The advantages of this environment:
  – It saves usually a team’s work to translate DFT function tests to STIL patterns in a project.
  – Avoids errors introduced in the manual translation process.
  – The generated pattern file can be 100% equivalent with simulation.
  – Can be applied to other formats that ATE needs.

• The disadvantage is, the STIL pattern cannot be obtained until the simulation finishes.
  – **How to resolve this?**
• We can add a fake run mode in the environment. In this mode, the DUT is replaced with a shell, which only has input/output pads defined, so that the pattern file can be obtained in minutes.