Scan chain based test setup for DE0-Nano based systems

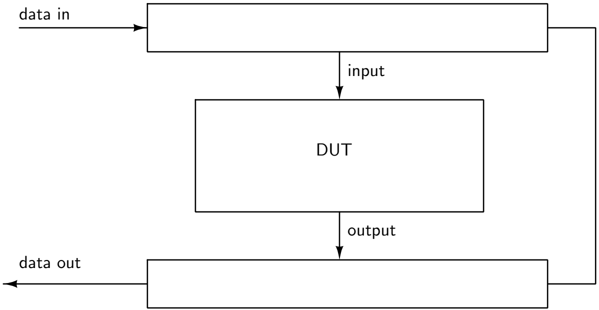
Titto Thomas

Wadhwani Electronics Laboratory
EE Dept. IIT Bombay

tittothomas@iitb.ac.in

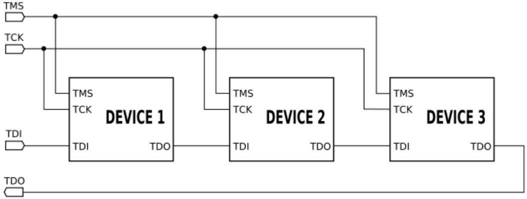
April 7, 2015

Scan Chain



- Scan chain is a technique used for testing the hardware systems.
- a simple way to set the inputs for the system and observe their outputs.
- It consists of two shift registers and their control signals.

Joint Test Action Group (JTAG)



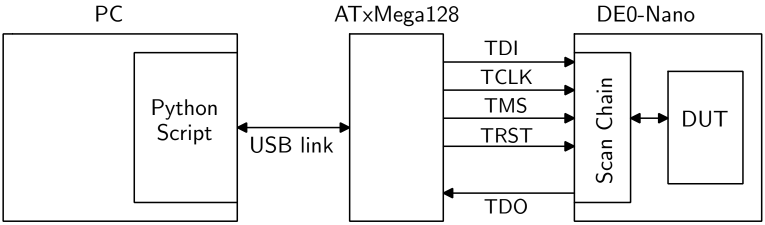
- Started as a method to test PCB boards, currently used as an industry standard for testing.
- It has a boundary scan architecture, i.e all the input and output pins are linked together in a set called the Boundary Scan chain.
- A simplified version of standard JTAG is proposed here for testing designs on the DE0-Nano board.

www.wikipedia.org/wiki/Joint_Test_Action_Group

Main blocks

- Has three main parts.
  - `Python Script` : Gets the command inputs from the user in a text file.
  - `Microcontroller(ATxMega128)` : Convert these commands into a set of signals for the DE0-Nano board.
  - `DE0-Nano board` : Contains both the DUT and the proposed scan chain.
- The PC communicates to the microcontroller through a USB link , with a predefined standard data transfer scheme.
- The microcontroller will translate the commands, and generate corresponding signals through it's port pins.

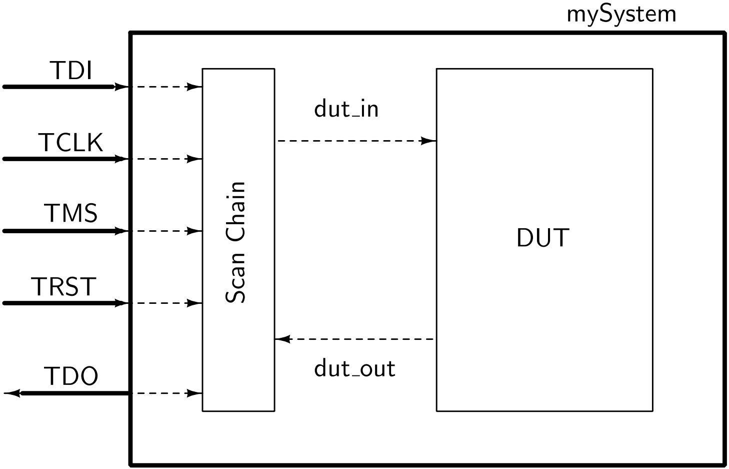
Block Diagram



Interface Signals

- The tester hardware contains the scan chain and the controller (TAP controller) that provides necessary control signals to it.
- The interface signals of this top level system would be
    - `TDI` : The serial test data input to be loaded in the scan chain.
    - `TMS` : The commands for the TAP (Test Access Port) controller are passed serially through this pin.
    - `TCLK` : The clock reference forin design for testing all the other communication lines.
    - `TRST` : Pin to reset the TAP controller at any instant.
    - `TDO` : The serial data output from the scan chain.

Adding scan chain



Adding scan chain (Contd.)

- DUT should first be tested in gate level simulation and verified to be working.
- user has to write a top level entity (shown as `mySystem`) which contains the DUT and `Scan_Chain` module as component.
- `mySystem` should have 5 interface signals, `TDI`, `TMS`, `TCLK`, `TRST` and `TDO`.

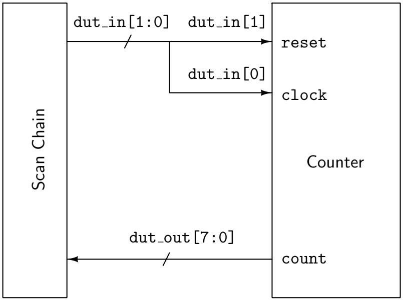
Scan Chain specifications

```vhdl
entity Scan_Chain is
  generic (
    in_pins : integer; -- Number of input pins
    out_pins : integer -- Number of output pins
  );
  port (
    TDI : in std_logic;  -- Test Data In
    TDO : out std_logic;  -- Test Data Out
    TMS : in std_logic;  -- TAP controller signal
    TCLK : in std_logic;  -- Test clock
    TRST : in std_logic;  -- Test reset
    dut_in : out std_logic_vector(in_pins-1 downto 0);
     -- Input for the DUT
    dut_out : in std_logic_vector(out_pins-1 downto 0);
     -- Output from the DUT
  );
end Scan_Chain;
```

Scan Chain specifications (Contd.)

- Scan chain has two configurable parameters (`in_pins` and `out_pins`) indicating the number of input and output bits to the DUT, which can be generic mapped.
- It also has one output (`dut_in`) and one input (`dut_out`) that should be connected to the DUT.
- Internally it contains an FSM (that implements the TAP Controller), one input scan register and one output scan register.

Reference Implementation : Counter



- The counter has two single bit inputs (clock and reset) and an 8 bit single output (count).
- Here, the dut_in will be 2 bits (one for each of the inputs) and dut_in will be same as count.
- The top level VHDL description of this system is given in supporting document.

Hardware connections

- Next step is to make physical connections between the host PC, microcontroller board and the user module (on DE0-Nano).
- The microcontroller board is PtX-128 (ATxMega128 based) developed in WEL lab, IITB. Connect it to the PC.
- The following connections between the microcontroller board and the DE0-Nano need to be made
    - `TRST` (DE0-Nano) to PD4 (PORTD.4)
    - `TDI` (DE0-Nano) to PD0 (PORTD.0)
    - `TMS` (DE0-Nano) to PD1 (PORTD.1)
    - `TCLK` (DE0-Nano) to PD5 (PORTD.5)
    - `TDO` (DE0-Nano) to PC0 (PORTC.0)

Input file format

- For testing the hardware, the user has to provide input combinations, their expected results and the time duration of execution.
- They should be written as commands in a text file and passed to the python script for test execution.
- These commands are derived from the Serial Vector Format (SVF), usually used in JTAG boundary scan.
- Only two commands are required for the current implementation.

SDR

`SDR` $<$ *in pins* $>$ `TDI(`$<$ *input* $>$`)` $<$ *out pins* $>$
`TDO(`$<$ *output* $>$`)` `MASK(`$<$ *mask bits* $>$`)`

- This Serial Data Register instruction is for carrying out a data scan in process.
- *in pins* & *out pins* contains the number of input and output bits respectively.
- *input* & *output* contains the input combination to be applied and it's expected output combination respectively.
- *mask bits* are used to specify if any of the output bits are not important and could be taken as don't care

Example : `SDR 2 TDI(0) 8 TDO(00) MASK(FF)`

Note : If the scanned output should not be compared, then all the *mask bits* should be kept as 0.

RUNTEST

`RUNTEST` < *delay* > `SEC`

- As the previous instruction loads the input and samples the output, this instruction is used to apply the input combination to the DUT and wait for *delay* seconds.

Example : `RUNTEST 60 SEC`

- The *input*, *output* and *mask bits* are to be written as hexadecimel numbers (uppercase for alphabets).
- An example input file is given in the supporting document.

Running the Python script

- First install pyUSB library on the PC by the following steps.
  - Download pyUSB v1.0 from the site "*http://sourceforge.net/projects/pyusb/*"
  - Follow the steps in the README document to install libusb and pyusb v1.0 on linux.
- Now run the `scan.py` script with the following command.

```
$ sudo scan.py <input file> <output file>
```

Where *input file* contains all the commands to be executed, and *output file* should be an empty file for storing the results.

Thank You