DFT Core Insertion for Digital Circuits using MICROWIND

Vinay Sharma
Director, ni2designs

Gireesh M.
Scientist, ISRO

Spurred by technological advancement, the number of devices that can be included in an integrated circuit is increasing every year. As the complexity and density are increasing, improved and efficient testing techniques are needed. Testing of complex integrated circuits is a tedious and time consuming task. Testing improves the quality of parts that are shipped. Testing devices in production is meant to screen out defective parts. Tests have to be generated that ensure a high probability of uncovering physical defects like shorts to ground or Vdd, bridges to neighboring signal lines, incorrect resistance of lines or via's and many more.

Testing costs are increasing because as the circuit grows in size and complexity, the controllability and observability of individual elements decreases, which lead to dramatically increased test generation time. To generate test patterns using ATPG (Automated Test Pattern Generator) algorithms and hardware penalty in Design for Testability (DFT) techniques are impediments to enhancing the fault coverage. To improve product quality, either test pattern generation algorithm or hardware overhead has to be tolerated. Out of these two, increasing testability by increasing area overhead is preferred, even-though it is at reduced performance, as it can result in more testable circuits. But industries are reluctant in adding additional hardware. Testing time and testing cost can be reduced greatly by selecting the appropriate DFT techniques.

This article deals with methods by which different DFT techniques can be added in an integrated circuit, at reduced test cost and testing time. Also this article introduces a *"DFT Core Generator"* software which is capable of generating JTAG cores for the digital circuit. The research with DFT core generator can be carried forward to develop cost effective test circuits and various DFT techniques.

Different DFT techniques have been used over the few past decades for testing IC's. Boundary Scan (BS) and Built in Self Test (BIST) are the most popular and industrially adopted DFT techniques. This article also deals with implementation of Boundary Scan (BS) technique and Built In Self Test.

Implementing IEEE 1149.1 (JTAG) Standard:
The IEEE 1149.1 Test Access Port and Boundary Scan Standard popularly called by the name JTAG (Joint Test Action Group) standard have been developed as an efficient standard for implementing boundary scan architecture. Due to increased complexity of integrated circuits, BIST is emerging as the state-of-art testing technique. BIST gets rid of the need for test pattern generation and reduced test overhead. In addition it enables high-speed testability.

This article discusses the techniques of developing a low cost Design for Testability (DFT) core generator which can be configured depending upon the design core that has to be tested. The DFT core include sub cores for JTAG controller for boundary scanning and Memory Built In Self Test (MBIST) controller for testing embedded RAM memory cells.

Boundary Scan Standard Concept
Boundary scanning testing is a general strategy used that allows controllability and observability of input/output pins of integrated circuits or die embedded in a system. The IEEE/ANSI 1149.1.1990 standard is a widely accepted industry standard for implementing boundary scan technique. The group that started working on this standard is called Joint Test Action Group (JTAG), thus this standard is widely known as JTAG standard.

In Boundary Scan architecture a memory cell is associated with every input and output pins of the IC. These memory cells, called boundary scan register, are serially connected forming a shift register. The JTAG standard also describes about a Test Access

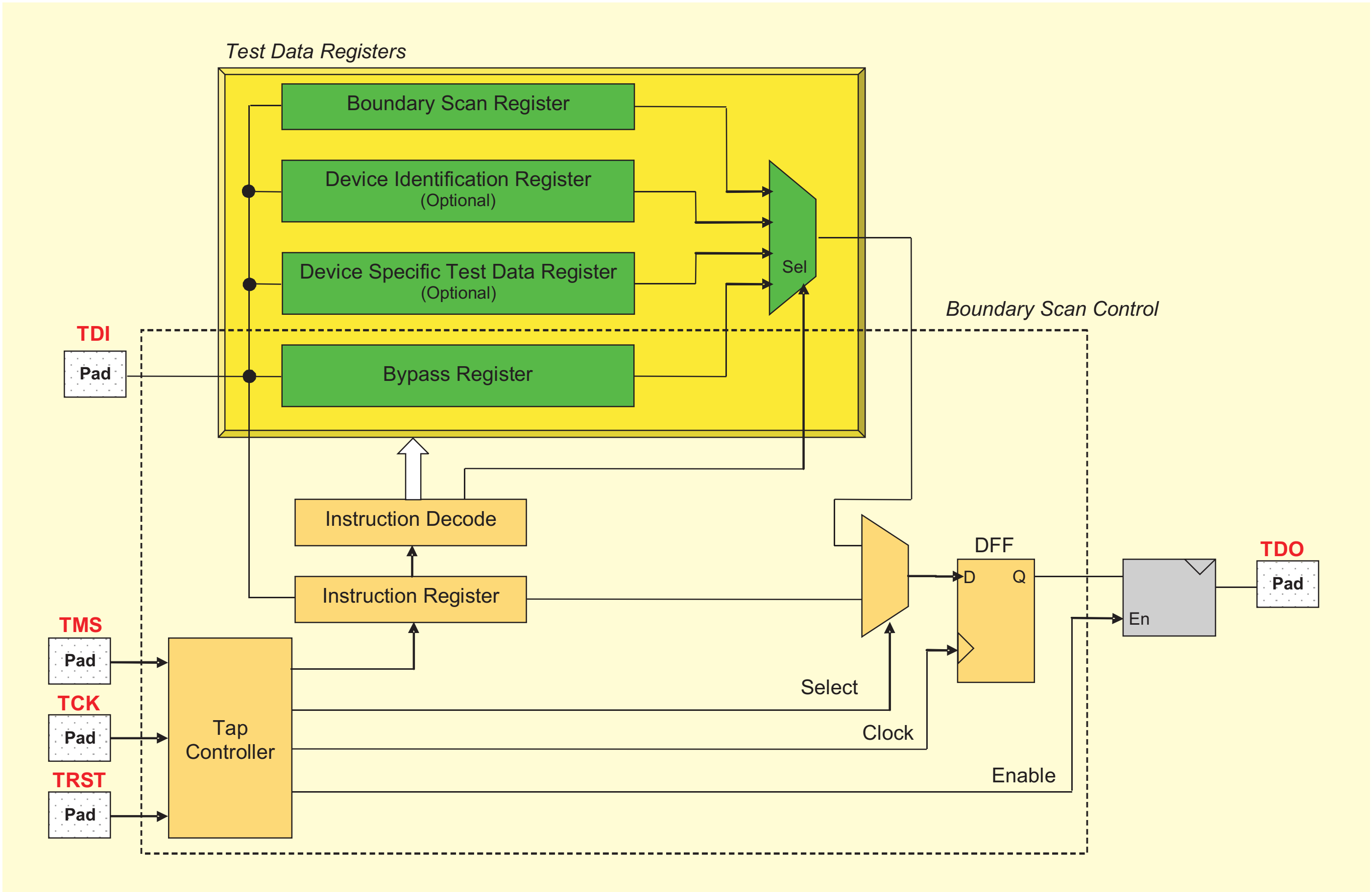


Figure 1: Boundary Scan Architecture Asynchronous Implementation.

Port (TAP) which provides test access to shift registers and controls the various chip test modes. The figure1 shows the basic architecture of JTAG standard.

Port (TAP) which provides test access to shift registers and controls the various chip test modes. The figure1 shows the basic architecture of JTAG standard.

The test bus circuitry of boundary scan technique consists of the boundary scan registers, 1-bit bypass registers, instruction register and decoder, some miscellaneous registers and Test Access Ports (TAP). TAP consists of five I/O pins. They are TDI (Test Data In), TDO (Test Data Out), TCK (Test Clock), TMS (Test Mode Select) and TRST (Test reset). Out of these TRST pin is optional. It is used for asynchronous reset in circuits where power on reset option is not provided. But it can also be used in circuits having power-on-reset. The test instruction and data are fed serially through the TDI pin. Test result and status information are send out from the chip through TDO pin.

The status of the test circuitry is decided by TAP controller. TAP controller is 16-state FSM (Finite State Machine). Status of a TAP controller is decided by the state transition on TMS pin and decoded at every positive transition of Test clock (TCK).

TAP controller is designed such that whenever the value at TMS pin is high for 5 successive test clocks, the TAP controller state will be transferred to the test logic rest state. The state diagram of TAP controller is shown in figure 2.

TAP controller generates eight different control signals - ShiftDR, ClockDR, UpdataeDR, ShiftIR, ClockIR, UpdataeIR, Select and Enable. During every state of TAP controller, corresponding control signals are generated. TAP controller consists of a instruction cycle and a data cycle. ShiftIR, ClockIR and UpdateIR control signals are used during instruction cycle and ShiftDR, ClockDR and UpdateDR are used during data cycle. Select control signal is used to select instruction register. Enable signal is used as enable signal for output buffers used during the execution of HIGHZ instruction.

Test Instructions:

JTAG standard specifies some instructions for testing as well as to access internal logic circuitry. The mandatory instructions that are specified by the standard are SAMPLE, PRELOAD, BYPASS and EXTEST. There are some optional instructions specified by the standard - IDCODE, INTEST, CLAMP, HIGHZ and RUNBIST. In addition provisions are given to add user defined instruction. Mainly testing of systems or devices are done through EXTEST, INTEST and RUNBIST instructions.

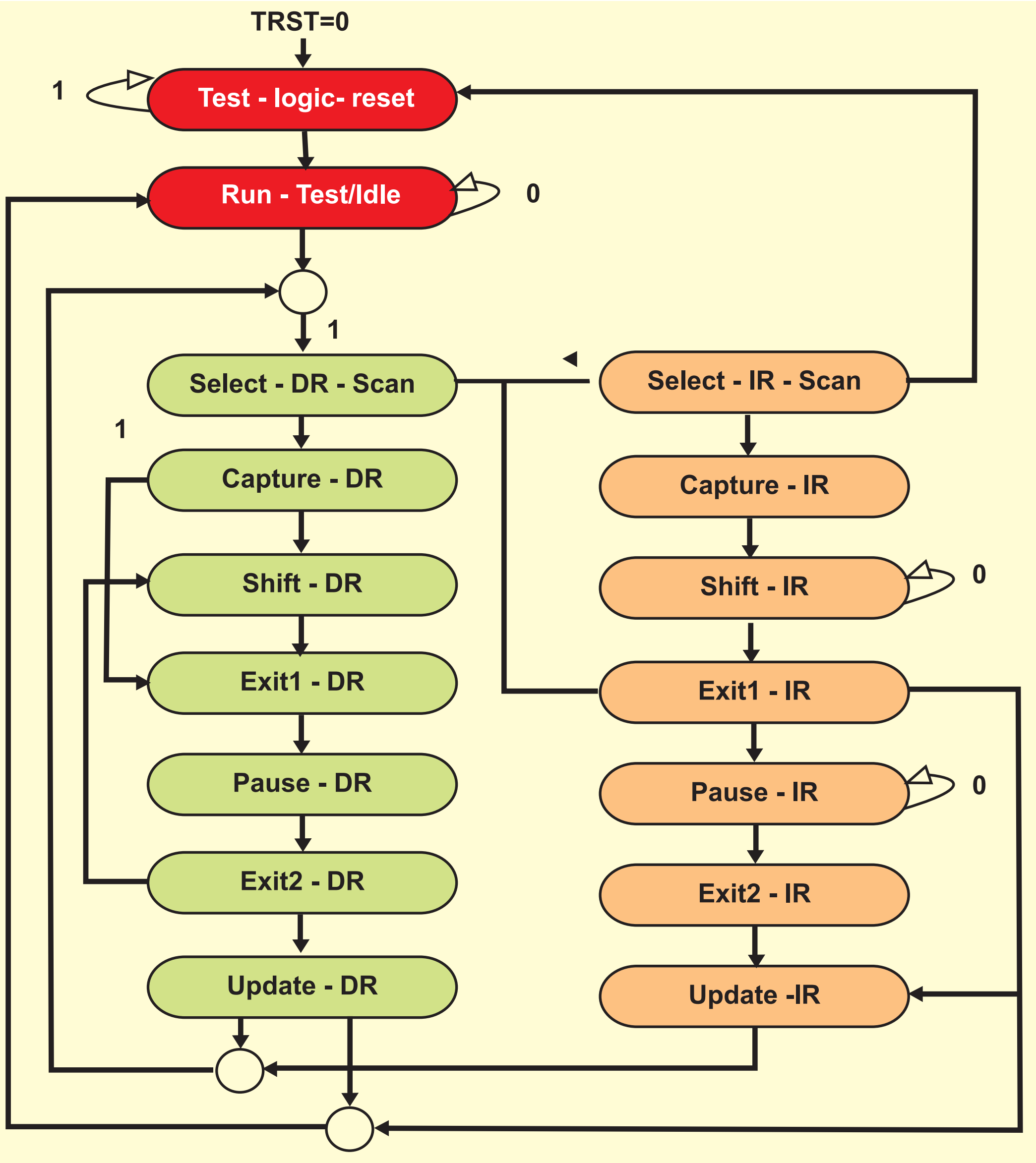


Figure 2: TAP Controller State Machine.

In this article & in our DFT core generator, RUNBIST instruction is used to control Memory BIST controller so that input pin reduction can be achieved. EXTEST instruction is mainly used for testing external board connectivity and is used generally for checking PCB's. For INTEST instruction the test pattern that are applied will flow through the internal shift register path other than the boundary scan path, to check the underlying circuitry of an integrated circuit.

The instructions are fed to instruction register through TDI pin. By default the instruction stored in the instruction register will be that of the IDCODE instruction, used for retrieving the JEDEC standard identification code, stored in the 32 bit long identification register. If identification register is not at all implemented then, IDCODE instruction code will be assigned to BYPASS instruction. The width of the instruction register is dependent on the number of instructions implemented. The instructions are loaded during the Shift-IR state of the TAP controller. The instructions loaded will be latched onto the instruction decoder only during the Update IR state of TAP controller. The instruction decoder decodes and selects the appropriate register.

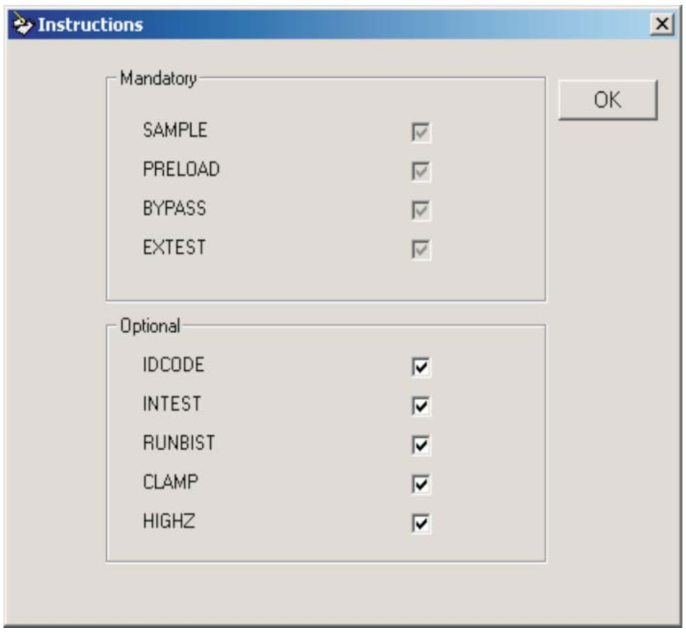
The width of the instruction register is not specified by the standard. It is only stated that it should not be less than 2 bit wide. The user can select the width depending upon the number of instructions used and the area overhead of the decoding circuitry. In this article, a 6 bit wide instruction register is used to implement 9 instructions. For implementing 9 instructions a 4 bit wide instruction register is sufficient, but the area occupied by the decoding circuitry is greater than the area overhead of a 6 bit wide instruction register. Six bit wide instruction register is used so that additional user defined instruction can be implemented with minimum area overhead.

The designer of an ASIC chip is free to select the different configuration of optional instructions specified by the standard. Depending upon the instructions selected the decoding logic is reconfigured. For the ease of reconfiguration 6 bit wide instruction register is selected. Figure 3 shows the list of instructions supported by DFT core generator.

Bypass instruction is provided to facilitate board level testing. In board level testing when ever a particular chip in the scanning chain has not to be tested, then bypass instructions is fed to that IC through TDI. All the instructions codes that are not used are assigned for BYPASS instruction

One output multiplexer is implemented to select from which of the registers the value has to be transferred to Test Data Output (TDO) pin. The select lines of the output multiplexer are controlled by the instruction decoder.

In this article since we are incorporating MBIST controller with TAP controller, BIST result also will be routed through TDO pin. The insertion of these components into the core logic is illustrated later in article.



Mandatory

SAMPLE ✓
PRELOAD ✓
BYPASS ✓
EXTEST ✓

Optional

IDCODE ✓
INTEST ✓
RUNBIST ✓
CLAMP ✓
HIGHZ ✓

Figure 3: Supported Test Instructions.

Memory Built In Self Test (MBIST) Concept
Most of the present day micro-electronic systems contain high density embedded memories. Besides production time testing, periodic testing of memories is essential to deal with problems such as data retention. Testing of memories using conventional techniques for microprocessors and digital communication equipment is not a feasible option. So testing techniques like Built In Self Test for memories are developed. Different types of MBIST architecture hardware centric MBIST, software centric MBIST and software/hardware centric MBIST are used. The block diagram of a Memory testing technique is illustrated in the figure 4.

Hardwired MBIST is selected even though it offers zero flexibility, it is due to its short testing time and low area overhead. The MBIST architecture used in this article uses MARCH-C algorithm for generating test pattern. MARCH-C algorithm is preferred over others due to its high fault coverage for both bit-oriented and word oriented memories. The block diagram of MBIST controller is shown in the figure 5.

The MBIST architecture consist of a clock divider, pattern generator, address generator and a comparator. Memory BIST operation is activated by RUNBIST instruction loaded into the TAP controller. The testing time of the MBIST operation depends upon the width and capacity of the embedded RAM. The same test clock is used by both JTAG controller and MBIST controller, for achieving reduced input pins.

The clock divider will divide the test clock (TCK) into four different clocks Read, Write, Address Clock and Compare. Read and Write control signals are connected to the memory block through a collar. Address Clock is used to drive the address generator,

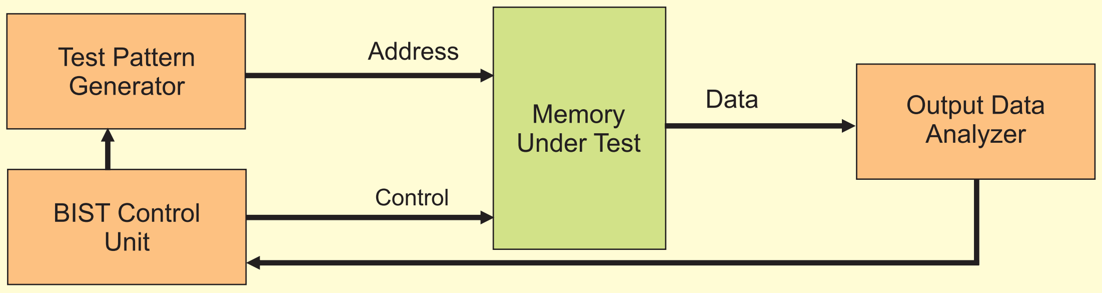


Figure 4: Block Diagram of BIST.

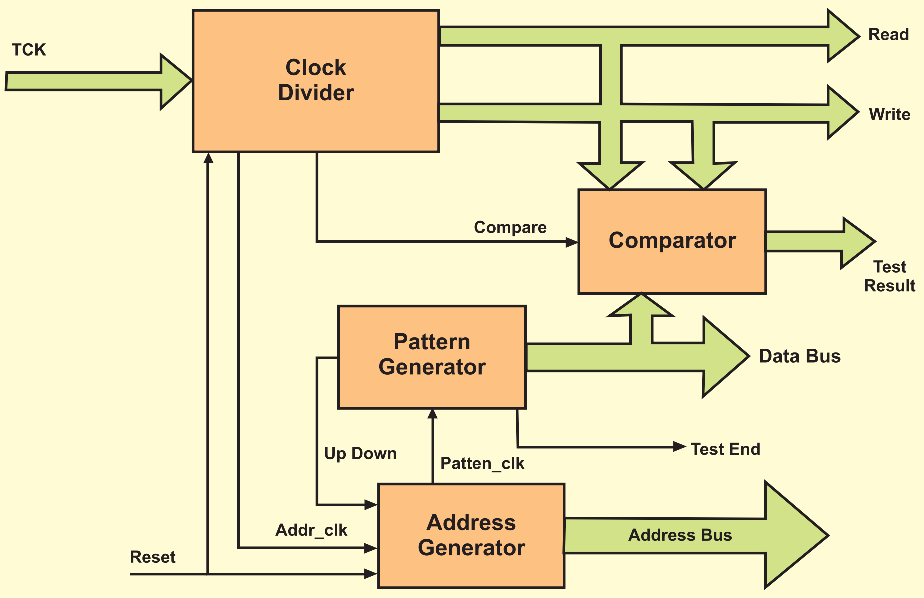


Figure 5: Block Diagram of MBIST Control

while compare clock is given to the comparator block of the architecture.

Collar is a set of multiplexers used for interfacing MBIST controller with the embedded memory. It used for multiplexing read and write control signals and data and address bus generated by the MBIST controller and the memory controller of the core logic.

Address generator is triggered by address clock address generator is made of a n-bit gray counter. Each test pattern has to be fed to all addresses in the memory. The gray counter is used to generate address because of area overhead and due to the fact that the address generated will not be of sequential order, which is necessary for detection of some fault models. The width of the address generator depends upon the width of the memory to be tested.

The pattern generator is triggered by a control signal pattern_clk generated by the address generator once all the memories are accessed. The pattern generator is also a address generator which points the address of LUT used to store the test pattern.
The test pattern is generated by the software which will be mentioned later. Pattern generator also uses a gray counter to locate the position of test pattern in the look up table. The depth of the LUT again depends upon the capacity of embedded memory. The test pattern is based on MARCH-C algorithm. Each test pattern will be given to each and every memory cell. Pattern generator also generates a control signal (UPDOWN) in addition to the test patterns. UPDOWN control signal is supplied to the address generator.

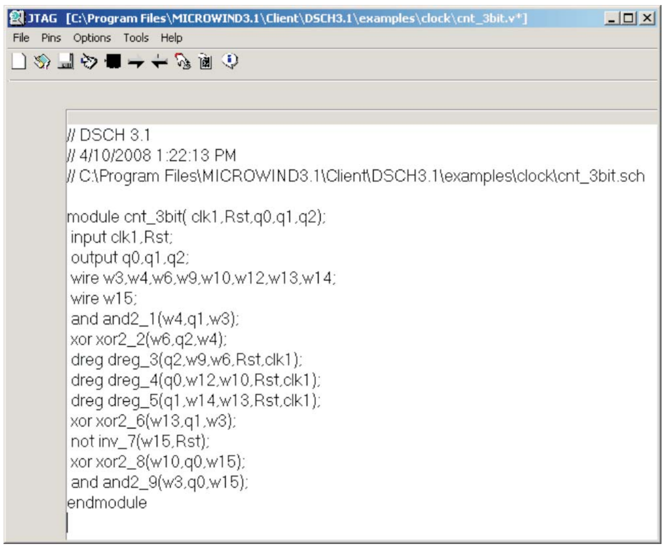
MARCH-C algorithms is preferred for generating test pattern because it can detect faults like stuck-at-fault (SAF), transition fault (TF), coupling fault (CF), neighborhood pattern sensitive fault(NPSF) and address fault (AF) with very high fault coverage. MARCH-C algorithm also decides the direction of memory flow.

Comparator is used as the output data analyzer. It consists of two register having the width of memory. One register is used to store the value of instantaneous test pattern. It is loaded while write control signal is applied. During read control signal content of the memory cell will be loaded into the second register. During compare phase, it will compare the contents of two register, and whenever a mismatch occurs, it makes Test Result signal high.

When ever the Test result goes high the testing process is halted. If no mismatch occurs during testing, then once all the test patterns are applied Test End signal goes high. When Test End signal goes high, the chip comes out of the memory testing process and it will be free for normal operation.

DFT Core Generator Software:
DFT Core generator or inserter are softwares which are capable of generating and insertion of DFT cores in digital circuits. To study and research more towards development of such software we designed our own DFT core generator for MICROWIND software.

The software offers intelligent insertion of boundary scan architecture and memory BIST controller into any existing switch level Verilog design code. It takes switch level Verilog code file as input and manipulates it in such a manner so that all the previously


JTAG [C:\Program Files\MICROWIND3.1\Client\DSCH3.1\examples\clock\cnt_3bit.v*]
File
Pins
Options
Tools
Help

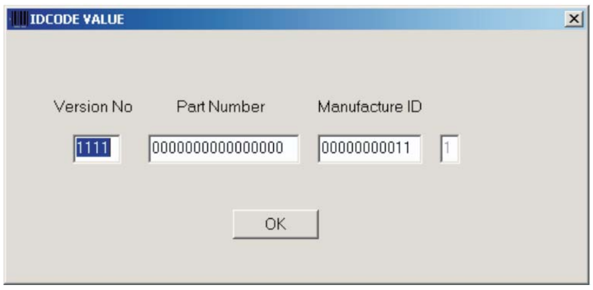
```
// DSCH 3.1
// 4/10/2008 1:22:13 PM
// C:\Program Files\MICROWIND3.1\Client\DSCH3.1\examples\clock\cnt_3bit.sch

module cnt_3bit( clk1,Rst,q0,q1,q2);
 input clk1,Rst;
 output q0,q1,q2;
 wire w3,w4,w6,w9,w10,w12,w13,w14;
 wire w15;
 and and2_1(w4,q1,w3);
 xor xor2_2(w6,q2,w4);
 dreg dreg_3(q2,w9,w6,Rst,clk1);
 dreg dreg_4(q0,w12,w10,Rst,clk1);
 dreg dreg_5(q1,w14,w13,Rst,clk1);
 xor xor2_6(w13,q1,w3);
 not inv_7(w15,Rst);
 xor xor2_8(w10,q0,w15);
 and and2_9(w3,q0,w15);
endmodule
```

Figure 6: GUI of DFT Core Generator.

discussed blocks are added appropriately. Figure 6 shows the main GUI of the DFT core generator.

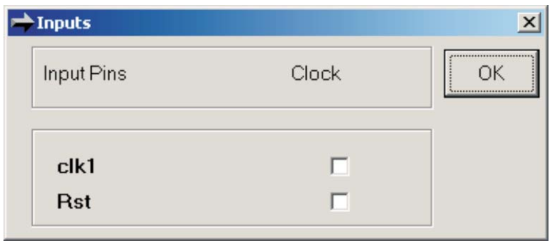
The software will detect the input and output pins from the input Verilog file. Once the input and output pins are detected, the appropriate boundary scan cells are inserted. Then the software

will ask for what all instructions are to be included. If IDCODE instruction is selected for inclusion, then the software will prompt (figure 7) for identification code. Once instructions and identification register is given, the program will insert the instruction register and instruction decoder depending upon the instruction selected. The software is also capable to identify all input nodes as shown in figure 8. Users can select and mark any input as clock signal used in the circuit. So once the generate button is pressed a complete DFT core is inserted in the main circuit and then it generates a Verilog description of the complete DFT circuit.
For inserting Memory BIST controller, the software prompt for


IDCODE VALUE

Version No	Part Number	Manufacture ID	
1111	0000000000000000	00000000011	1

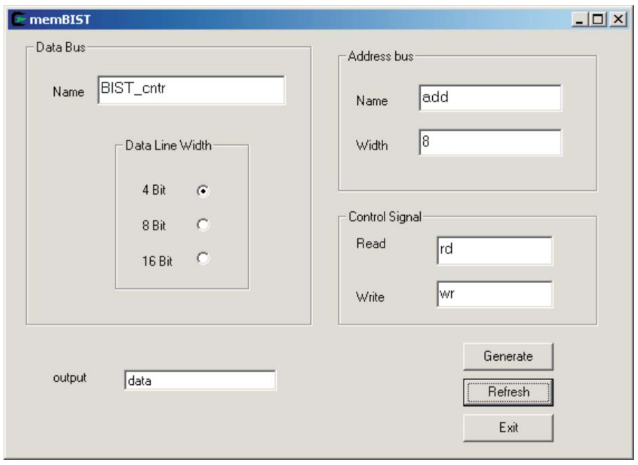
Figure 7: ID code Insertion.


Inputs

Input Pins	Clock
clk1	☐
Rst	☐

Figure 8: Detection and Selection of Clocks.

whether any memory block is inserted. If any memory block is present, then user must indicate the width and depth of memory block and the names specified for indicating control signals. The user must also specify the name of address bus and data bus. The user should select RUNBIST instruction, whenever a memory block is in the design. Figure 9 shows the GUI of the Memory BIST generator.


memBIST

Data Bus
Name: BIST_cntr
Data Line Width: [✓ 4 Bit] 8 Bit 16 Bit

Address bus
Name: add
Width: 8

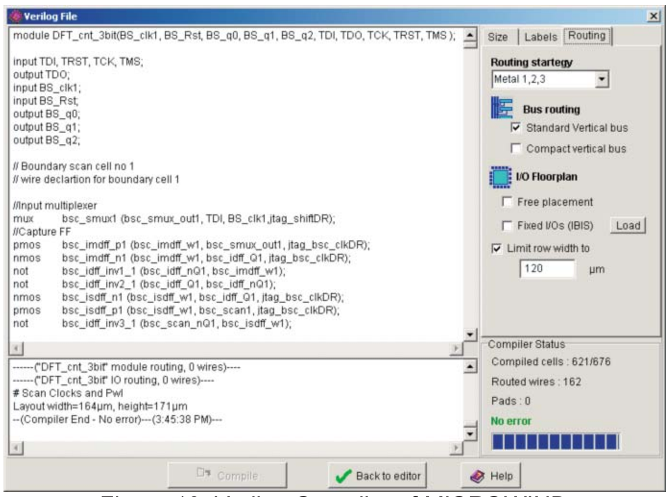
Control Signal
Read: rd
Write: wr

output: data

Figure 9: Memory BIST Generator GUI.

Implementing DFT Core using MICROWIND
The DFT core generator gives a switch level Verilog code which can be compiled using Verilog compiler of MICROWIND software, to see the layout of the core logic with DFT facility incorporated.
We used MICROWIND 3.1 with 0.12 um technology for design and testing of the generated Verilog circuit file. The results were to the satisfactory levels, with very low development time.

Figure 10 shows the DFT core's Verilog circuit compiled completely without errors using the layout compiler of


Verilog File

```
module DFT_cnt_3bit(BS_clk1, BS_Rst, BS_q0, BS_q1, BS_q2, TDI, TDO, TCK, TRST, TMS );

input TDI, TRST, TCK, TMS;
output TDO;
input BS_clk1;
input BS_Rst;
output BS_q0;
output BS_q1;
output BS_q2;

// Boundary scan cell no 1
// wire declartion for boundary cell 1

//Input multiplexer
mux       bsc_smux1 (bsc_smux_out1, TDI, BS_clk1,jtag_shiftDR);
//Capture FF
pmos      bsc_imdff_p1 (bsc_imdff_w1, bsc_smux_out1, jtag_bsc_clkDR);
nmos      bsc_imdff_n1 (bsc_imdff_w1, bsc_idff_Q1, jtag_bsc_clkDR);
not       bsc_idff_inv1_1 (bsc_idff_nQ1, bsc_imdff_w1);
not       bsc_idff_inv2_1 (bsc_idff_Q1, bsc_idff_nQ1);
nmos      bsc_isdff_n1 (bsc_isdff_w1, bsc_idff_Q1, jtag_bsc_clkDR);
pmos      bsc_isdff_p1 (bsc_isdff_w1, bsc_scan1, jtag_bsc_clkDR);
not       bsc_idff_inv3_1 (bsc_scan_nQ1, bsc_isdff_w1);
```

```
------("DFT_cnt_3bit" module routing, 0 wires)----
------("DFT_cnt_3bit" IO routing, 0 wires)----
# Scan Clocks and Pwl
Layout width=164µm, height=171µm
--(Compiler End - No error)---(3:45:38 PM)---
```

Routing startegy: Metal 1,2,3
Bus routing: [✓ Standard Vertical bus] Compact vertical bus
I/O Floorplan: Free placement, Fixed I/Os (IBIS)
[✓ Limit row width to] 120 µm

Compiler Status
Compiled cells : 621/676
Routed wires : 162
Pads : 0
No error

Figure 10: Verilog Compiler of MICROWIND.

MICROWIND. The MICROWIND's Verilog layout compiler offers control over the row width of the layout and selection on number of routing layers, transistor sizes, I/O pad bonding, etc.

The layout generated from the Verilog compiler looks like in figure 11. All transistors are laid out uniformly and routing is done automatically by the MICROWIND software.

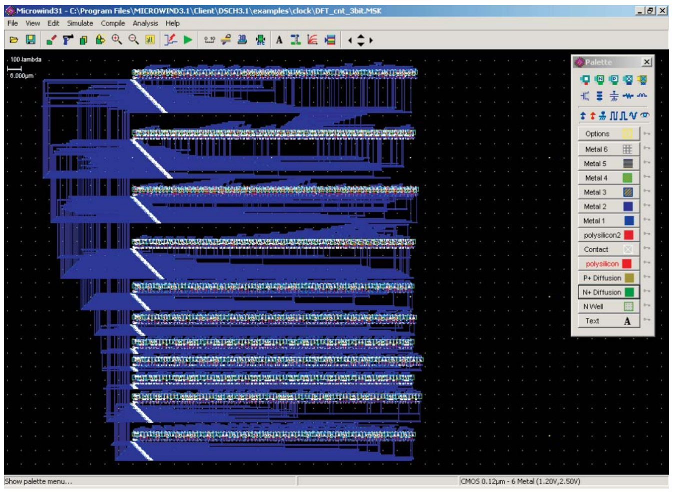


Figure 11: CMOS layout of DFT Core Generator.

Summary
The software developed for inserting Design for Testability (DFT) Core provides the designer with a convenient and reliable method for testing their digital circuitry. The software automatically inserts the JTAG and/or MEMBIST circuitry to any ASIC design using lesser number of transistors. It used 1020 nMOS and 1014 pMOS transistors for this design.

DFT core generator motivates researchers and learning engineers to work towards area of testability and come with more solutions.

For more information about MICROWIND software, please visit www.microwind.net