Board Level IEEE1149.1 Boundary Scan Built In Self Test

By
Stephen Harrison: Motorola steve.harrison@motorola.com
Peter Horwood: Firecron Ltd Sales@firecron.com

Abstract—*IEEE1149.1 Boundary Scan has become an important test technique within complex IC's and boards in today's electronic assemblies, providing a low cost, high fault coverage test methodology for digital designs.*
The most common approach is for the IEEE1149.1 test to be performed in factory with test vectors being supplied by external test equipment, however new IEEE1149.1 test support devices are now becoming available that support enhanced IEEE1149.1 test solutions, by enabling system and field use of IEEE1149.1, while also embedding IEEE1149.1 test capability within a design.
This paper shows how the IEEE1149.1 tests developed for factory test can be embedded and reused within a product, providing a high quality GO/NOGO built in self test (BIST) capability during the entire products life cycle.
This embedded IEEE1149.1 BIST technique will be demonstrated by showing an implementation within a complex real life telecommunication product; it will take the readers through the basic hardware and software design requirements.

STANDARD IEEE1149.1 TEST

The Boundary-Scan tests prime function is to establish that all Jtag testable digital connections on an electronic assembly are free of faults, a typical test sequence would consist of five functions: "*Infrastructure*", *"Interconnect*", *"Cluster*", *"Device BIST*", *"Device Programming*".

Infrastructure

The first Boundary-Scan test normally performed is a scan-chain "*Infrastructure*" check, this establishes that all the standard IEEE1149.1 bus signals are free from manufacturing defects and operate correctly *(TDI, TDO, TCK, TMS & TRST)* and forms the solid base from which all-further tests are performed.

Interconnect

The *"Interconnect*" test performs an open and short circuit test between all circuit traces on devices that support IEEE1149.1, this would typically includes paths through simple components that can be treated as transparent, i.e. resistors and buffers.

Cluster test

A cluster is defined as a portion of a design that can be totally accessed and controlled via IEEE1149.1 compliant devices. The *"Cluster*" test would normally perform an interconnection and stuck at test of address lines, data lines and control lines of a specific cluster, such as SRAM, DRAM Dual-Port etc.

Device Level BIST

Device "*BIST*" as its name describes, performs an internal device level test of the silicon, this type of test is normally seen on larger application specific integrated circuits (ASIC) devices, although some smaller devices now have this function.

Device Programming

Its common practice within product manufacturing for cPLD's & FLASH memories to be programmed using IEEE1149.1.

EMBEDDING THE TESTER

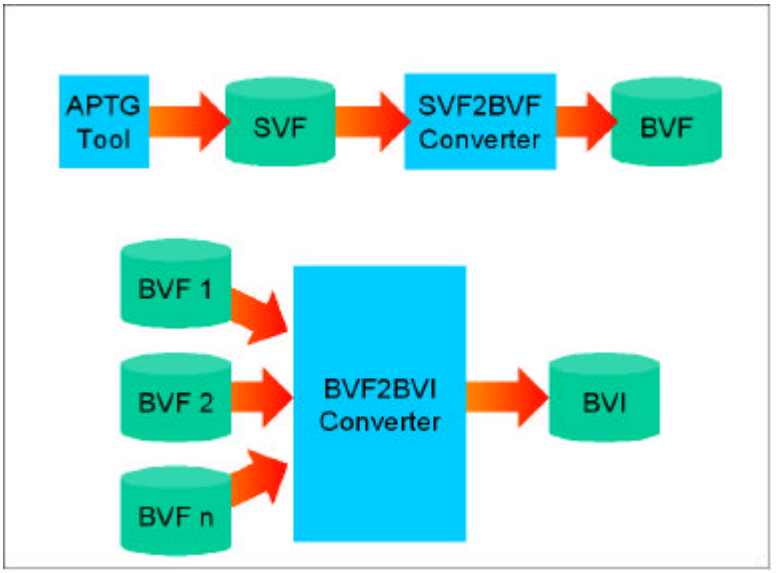
The decision to embed a tester will depend on each company's approach/strategy to its products ability to self-test, with cost, hardware and software engineering playing a major factor, each of these need to be addressed and a decision made. Motorola's aim was to develop with a third party an IEEE1149.1 BIST strategy that could be implemented simply without the need for an on board microprocessors to control the BIST sequence.
Motorola chose its 3G digital modem product to implement IEEE1149.1 BIST as it's IEEE1149.1 architecture consisted of multiple scan chains controlled via a multi-drop, hierarchical bridge device. The 6 scan chains contain a large number of cPLD's, FBGA's, PowerPC's and ASIC's, with each ASIC having its own internal device BIST.
The goal of the IEEE1149.1 BIST sequencer was to execute board level and device level test vectors on board without a access to external test equipment, the test vectors would be ran from several different scenarios;

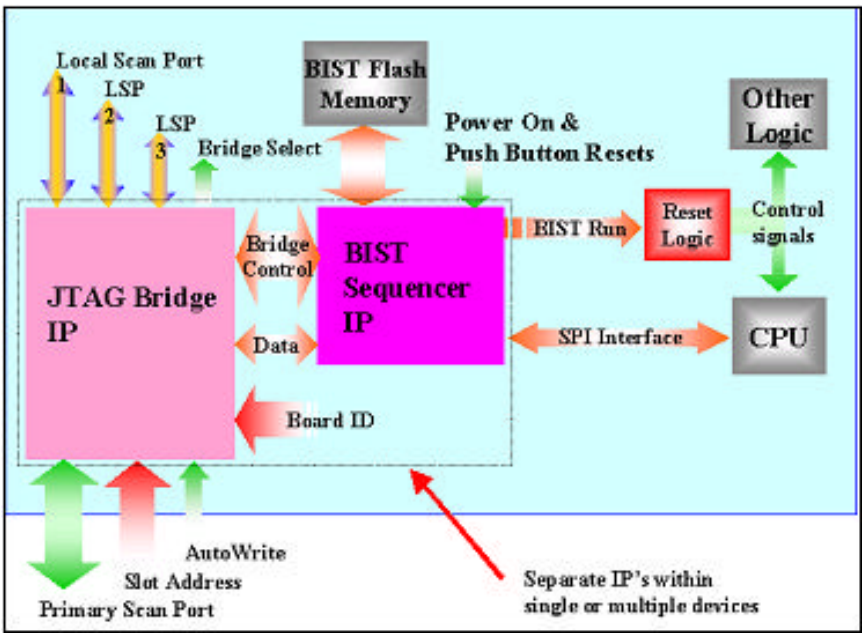
a) On power up.
b) Software reset
c) Hardware reset
d) By software request

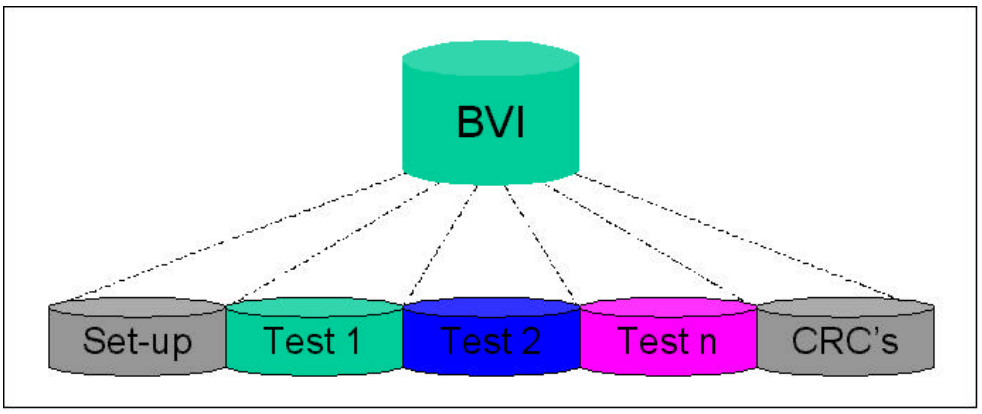
The hardware solutions chosen to implement this initially consisted of an IP core within a cPLD, with external Flash memory for test vector storage, however the IP core has now migrated into a commercially available device.
The features the BIST sequencer supports are:

1) Run test sequence initiated on power up.
2) Run test sequence by software command.
3) Provide a GO/NOGO flag.
4) Ability to change TCK rate via stored vectors.
5) Ability to execute, compare and store multiple test sequence CRC's.
6) Have the ability for a microprocessor to access the CRC results for further analysis.
7) Have the ability for a microprocessor to re-program the IEEE1149.1 BIST storage Flash with new vectors.
8) The ability to "*Hold Off*"/Control other devices within the design.
9) Ability to support multiple Flash memory types.

The hardware implementation chosen for the 3G design has the BIST sequencer attached to flash memory for test vector and result storage.







Test Vectors

The choice of which test vectors to embed rely on them being:

1) 100% reliable with no ambiguous results.
2) Vector set size.

Another consideration to take into account is the level/granularity of diagnostic required; this will be explained in more detail later.

The BIST sequencer reuses test vectors developed using standard automatic IEEE1149.1 test program generation tools that can be output/exported in serial vector format (SVF). The SVF files are then converted into a compressed binary vector format (BVF) for use with the BIST Sequencer.

Binary Vector Format (BVF)

A BVF file is generated using the SVF2BVF utility, this parses and compresses an SVF file generated from any APTG tool. The BVF format is an intermediate step prior to generating an optimised 16-bit wide flash memory image.

Binary Vector Image (BVI)

After single or multiple SVF files have been converted into their individual BVF files, they need to be further processed into a Binary Vector Image (BVI) suitable for storing within flash memory. This additional processing is performed by the BVF2BVI utility; its function is to concatenate multiple BVF files into a single BVI file, generate individual CRC variables for each of the concatenated tests and add specific sequencer commands to vary TCK rate etc.
The generated CRC's are then used to determine pass or fail status during a BIST sequencer run.

Vector Compression

The vector compression ratio determines the size of the Flash memory required and is dependant on the number of SVF test concatenated within the BVI.

As a guide, a typical BV1 compressed "*Interconnect*" test for a complex design with thousands of components occupies approximately 800KB, where as "*Infrastructure*" test and "*Device BIST*" BVI can be as small as 1-10K. The table below shows some more typical BVI compression rates.

Format	Infra	Inter	Device BIST
APTG Tool	1K	2.9K	N/A
ATPG SVF	0.5K	1.4K	2.0K
BVI	0.15K	0.6K	0.2K

Note: - ATPG size is for internal ASCI file formats.

In order to provide storage for a full factory inter connect, infrastructure and BIST test for a high complex PCB, the total Flash memory size would be 16Mbit (TE28F160). However this could be reduced to 8 or 4 M-bit for infield/system confidence tests.

Vector CRC

As mentioned above each SVF/BVF generates a CRC, which is stored within the last segment of Flash memory. The software algorithm used to generate this is also hardware instantiated within the BIST sequencer silicon, once a test is initiated via the BIST sequencer the received TDO data stream generates an internal CRC signature, once complete the BIST sequencer will store and compare the CRC with the software generated CRC signature, setting a pass/fail flag within the BIST sequencer appropriately.

Programming the BIST Flash

Once the BVI file has been created it must be programmed into the flash device, this can be performed in several ways. The quickest is to use the embedded IEEE1149.1 component within the BIST sequencer, however it can also be done via the I^2C or SPI interfaces. The SPI & I^2C interfaces provide the facility to locally update and verify the flash image from the local processor fitted to the PCB.

BIST Execution Commands

The BVF file supports various commands to set-up and initiate action within the BIST sequencer, the major system commands are:
"GO": Starts sequencer execution of converted test vectors generated by third party APTG tools.
"OSC_DIV": Sets the clock divide ratio for the incoming oscillator to generate the system TCK.
"COMPARE": Compares the LFSR signature generated by the BIST sequencer when running a test to the known good value within BIST Flash, it also sets the error flag if an error present.
"END": Ends the test sequencer execution.

BIST Sequencer Hardware

The BIST Sequencer hardware has been implemented together with a 6-port IEEE1149.1 bridge in a single 256 I/O 1-mm pitch 17x17-mm BGA device. The logic implements all the required control, read and write access to the flash memory as well as providing GO/NOGO status flag. (See diagram below)

The following section describes in more detail the sequencer I/O

Sequencer_STATUS(0)
This output signal when high indicates that the sequencer is executing tests from the Flash memory. No access to the primary IEEE1149.1 port is possible when this signal is asserted.

Sequencer_STATUS(1)
This output signal indicates the completion status of the BIST tests executed by the sequencer from Flash, a HIGH indicates that the tests have failed and a LOW a pass

Sequencer_OSC: This input provides the master BIST clock for the sequencer operation

Sequencer_RST: This active low input signal resets the sequencer and then initiates the execution of tests from the Flash device

Sequencer_RUN_IN: This active low input signal initiates the execution of the Flash tests with out resetting the sequencer

Sequencer_RST_OUT: This output signal active low indicates that the execution of tests via the FLASH was initiated by the Sequencer_RST signal

Sequencer_RUN_OUT: This output signal when low indicates that the sequencer device is executing tests from the Flash memory

The BIST Sequencer has been designed to communicate directly with 16 Bit wide data bus, utilising the Intel CUI method of controlling the Flash write operations

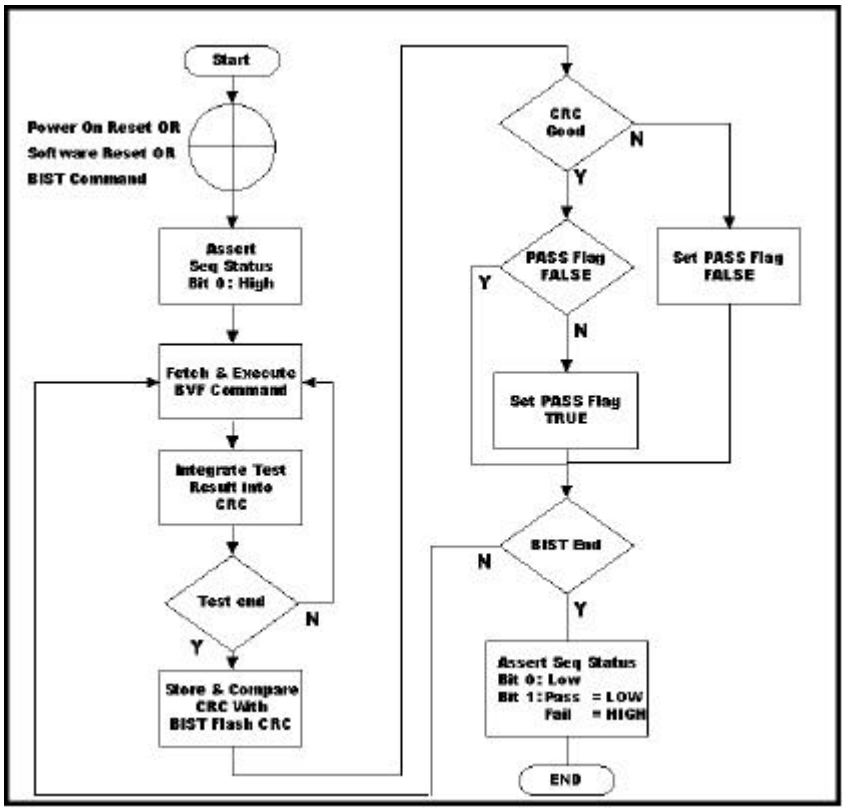
Sequencer_FLASH_RD: This active low output signal provides the read signal to the BIST Flash memory used to store the test vectors.

Sequencer_FLASH_WR: This active low output signal provides the write signal to the BIST Flash memory

Sequencer_FADD: These output signals provide the address bus to the BIST Flash device.

Sequencer_FDB: These bi-directional signals provide the data bus to exchange data with the BIST Flash.

I2C& SPI is available to provide the microprocessor with the ability to communicate with the BIST Sequencer, utilising this interface the microprocessor is able to read or write to the Flash memory and also initiate a run of the BIST Sequencer from a commanded memory location, this allows different test sequences to be stored within the Flash and run on command. The microprocessor is able to read from the Flash memory to identify, which test failed and using this information determine if the PCB can be brought into service using redundant sections of logic.



Sequencer Test Flow

The BIST test sequence is initiated via three possible scenarios, "Power On Reset", "Software Reset" & "Software Command", once this has taken place the device asserts the sequencer status bit (0) HIGH during the length of the BIST sequence. Once the status bit is set the sequencer then reads and executes the first BVF memory location. The typical sequencer flow can be seen in the diagram below.

Test Segmentation & Selection

The segmentation and selection of the test to be embedded should be considered carefully, it was our initial aim to reuse the code developed for factory test, however this proved to be unsatisfactory due to the level of diagnostics given, when running your test vectors in factory you have access to the debug and analysis tools of your APTG vendor, when embedded this is not the case. It was therefore necessary to split the test down to smaller segments each with its own CRC, this improved the granularity of the diagnostics provided.
The test was sequence was split down to the following elements.

- Infrastructure:
- ASIC 1 BIST
- ASIC 2 BIST
- ASIC n BIST
- ASIC n External Memory Test
- SRAM 1 test
- Etc, etc

Other items that should be carefully considered are the state of any off board I/O when the BIST sequence is running as this could cause system issues and or catastrophic system failures.

Test Run Time

Due to the manner in which the sequencer operates i.e.: - single or multiple tests run time is completely under the control of the user, the more segmented test are present, the more the sequencer has to access the flash to compare CRC's. The test can typically run faster than with a dedicated tester due to signal quality. As a bench mark a typical PCB infrastructure will take micro seconds where a full factory inter connect test for a highly complex PCB with over 3K components and many thousands of nets will take approx 4 seconds.

FURTHER POSSIBILITIES

FPGA-programming

Another task that must be performed within a system that incorporates FPGA devices is the loading of the devices programmable image. Currently there are 2 techniques favoured for device configuration, these are to store the FPGA image within Flash memory used by the microprocessor, which enables a parallel load of the data under control of the microprocessor, the second is to use IEEE1149.1 serial EEPROM's to hold the FPGA configuration data. These EEPROM's are not mainstream flash devices, but are specialised 1 bit wide devices and hence their cost per bit of storage is more when compared to normal flash memory.
The sequencer can provide a cost effective solution for the replacement of the custom EEPROM's, by incorporating the FPGA programming as part of the sequence of tests, that runs automatically upon power on. This involves obtaining an SVF file of the FPGA image and processing through the Firecron tools in the same manner as any standard test. The advantage is that the PCB can be fully tested and FPGA's programmed avoiding high cost serial EEPROM devices by utilizing the sequencers industry standard word wide Flash devices.

ACKNOWLEDGMENTS

We would like to thank Eugene Mullen of Firecron Limited UK for his invaluable assistance in supporting the IP core development.
Also Alan Moore and Greg Noeninckx within Motorola for there development and support activities.

Abbreviations

3G:	Third Generation
APTG:	Automatic Program Test Generator
ASIC:	Application Specific Integrated Circuit
BIST:	Built In Self-Test
BVF:	Binary Vector Format
BVI:	Binary Vector Image
CPLD:	Complex Programmable Logic Device
CRC:	Cyclic Redundancy Check
DRAM:	Dynamic Random Access memory
EEPROM:	Electrically Erasable Programmable Read Only memory
FPGA:	Field programmable Gate Array
IC:	Integrated Circuit
IP:	Intellectual Property
LFSR:	Linear Feedback Shift Register
PCB:	Printed Circuit Board
SRAM:	Static Random Access Memory
SVF:	Serial Vector Format