Leveraging Boundary Scan resources for comprehensive cluster testing

Heiko Ehrenberg and Norbert Muench

GOEPEL Electronics LLC
9600 Great Hills Trail, 150W, Austin, Texas 78759 USA

Abstract

Board level Boundary Scan testing as defined in IEEE-1149.1 is well established in the electronics industry. To achieve the best possible test coverage and testability for a specific design, a thorough DfT[1] Analysis is required though. DfT for Boundary Scan does not just include routing the scan chain and taking care of compliance pins on Boundary Scan devices. It also concerns non-Boundary Scan circuitry such as logic or memory clusters. Modern memory devices, especially synchronous memories, become more complex and faster with each generation. This paper suggests ways to ensure that those memory clusters will still be testable via Boundary Scan. Also, alternatives to memory cluster tests will be discussed, such as BIST[2] and test resources embedded in the memory device.

1. Introduction

The purpose of this paper is to provide the reader with an overview of the capabilities and limitations of various board and system level Boundary Scan (a.k.a. IEEE-Std.1149.1, JTAG) applications common for modern electronics. Test applications such as Interconnect Testing (Continuity Testing) between Boundary Scan I/O pins and between Boundary Scan and non-Boundary Scan I/O pins will be discussed as well as In-System Configuration (ISC) applications for EEPROM[3] and PLD[4]/FPGA[5] devices. We will focus especially on board level memory cluster test applications. Topic related Design-For-Testability guidelines will be interwoven throughout the paper. The basics of the IEEE-1149.1 standard will not be discussed in this paper. Rather, readers not familiar with the subject are referred to [1] and [2].

IEEE-1149.1 (referred to as Boundary Scan throughout this paper) describes a quasi static test methodology for digital circuits, mainly used for structural testing of board and system level connectivity. The standard does not provide for the measurement of PCB[6] trace or component parameters, such as resistance, capacity, inductivity, oscillator frequency, power consumption, etc. IEEE-Std.1149.4 has been developed to cope with analog and mixed signal test applications at board level. In addition to IEEE-1149.1 resources, components compliant to IEEE-1149.4 provide special test resources for the injection of current to and the measurement of voltage levels at circuit nodes. [3] [4]

Especially in the telecommunications and the networking industry, high-speed interconnects running at a data rate of multiple Gigahertz are common today. Endpoints on such interconnects are typically AC coupled. IEEE-1149.1 does not provide for the test of such signal paths. Also, IEEE-1149.1 has limitations in the test of differential lines. To cope with such limitations, the new IEEE-Std. 1149.6 has been developed. [5] [6]

2. Boundary Scan Interconnect Test

2.1 Board level interconnect test

Typically, today's electronic designs include one or more devices that provide testability features compliant to IEEE-1149.1. If the board designer provided the means to access the Test Access Port of such devices, ideally by creating a Boundary Scan chain (Figure 1), then the interconnections between Boundary Scan I/O's can be verified. The goal of such an Interconnect Test is to detect and - if possible - diagnose structural faults such as stuck-at1/0 or shorted nets and open pins.

Interconnect Tests are executed while the Boundary Scan devices involved are in EXTEST[7] mode. The test sequence includes preloading of Boundary Scan cells (input and output cells as well as control cells) and subsequent capturing of net logic levels in input cells. The IEEE-1149.1 (Boundary Scan) test resources

[1] DfT: Design for Testability
[2] BIST: Built-In Self Test
[3] EEPROM: Electrical Erasable Programmable Read Only Memory
[4] PLD: Programmable Logic Device
[5] FPGA: Field Programmable Gate Array
[6] PCB: Printed Circuit Board
[7] EXTEST: pin permission Boundary Scan instruction defined in IEEE-1149.1-2001, section 8.8

available on a net determine the degree of testability for structural faults on that net. Figure 2 provides a few examples of nets with different amounts of Boundary Scan test resources.

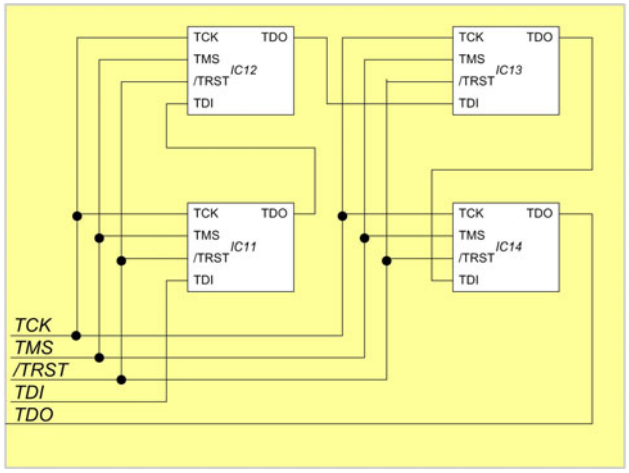


Figure 1: Board level scan chain; ring configuration

Figure 2-A represents a net that includes a pin with a Boundary Scan output cell only. The output cannot be deactivated. Furthermore, there is no Boundary Scan input cell available. Thus, open or stuck-at faults on such a net cannot be detected via Boundary Scan. To be able to test the net for such faults, additional resources (at least one Boundary Scan input cell) need to be provided, for example by means of external test modules connected to the net under test through a test point or peripheral connector pin. A short between a net with an output cell only and another net can only be detected if the other net provides at least a Boundary Scan input cell.

In Figure 2-B the only Boundary Scan pin available provides an input cell. It does not have any output capability. On a net like this the testability via Boundary Scan is limited to the detection of a stuck-at fault (partially) or an open, if the net provides a pull resistor. For example, if the net has a pull down resistor, a Boundary Scan test can detect a stuck-at high fault, but not a stuck-at-low fault. An open can only be detected in this case, if the open pin features internal pull-up circuitry. For such a test, the input cell would be preloaded with logic value High (Shift-DR state). When the TAP Controller[8] is passing through the Capture-DR state, the input cell should capture a Low (due to the pull-down resistor on the net). If the cell does not capture a Low, the pin seems open or stuck-at high, or the pull-down resistor may be missing. To detect a short between the net shown in Figure 2-B and another net, this other net must provide at least a Boundary Scan output cell.

[8] TAP Controller: Test Access Port Controller (state machine), defined in IEEE-1149.1-2001, section 6

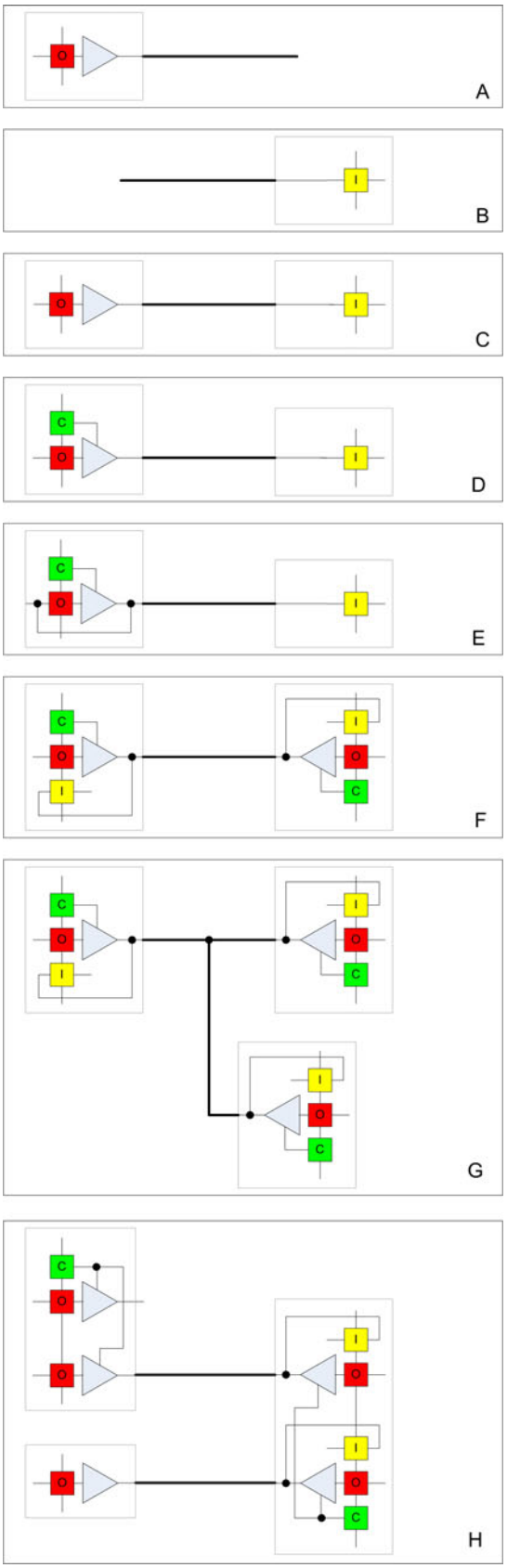


Figure 2: Nets with various levels of testability

The net shown in Figure 2-C includes one Boundary Scan output cell and one Boundary Scan input cell.

Figure 2-D represents the connection between a Boundary Scan output cell that can be deactivated and a Boundary Scan input cell.

In Figure 2-E, the Boundary Scan output cell can be deactivated and is self-monitoring. The other Boundary Scan cell in this net is an input cell.

The net in Figure 2-F includes two bi-directional Boundary Scan pins, each with individual cells for control, input data, and output data.

The Boundary Scan resources available in those nets pictured in Figures 2-C through 2-F allow the detection of opens, stuck-at faults and shorts, however, opens cannot be localized down to the pin level (either of the two pins could be open). If the net shown in Figure 2-F would be shorted to another net and at the same time one of the two Boundary Scan pins in net 2-F would be open, that open pin could be localized, though.

Figure 2-G shows a net with three bi-directional Boundary Scan pins with three cells each (Control, Input, and Output). Such a net would be fully testable. Furthermore, if only one pin is open, that pin could be localized.

In Figure 2-H, some of the Boundary Scan pins use shared control cells. Shared control cells limit the control over the involved Boundary Scan pins, since those pins can only be activated or deactivated together. E.g., in this example the two Boundary Scan output pins on the upper left side of the circuitry share one control cell. The two bi-directional Boundary Scan pins on the right also share a control cell. An output only Boundary Scan pin on a third device is connected to one of the bi-directional pins. Since that output pin cannot be deactivated, the bi-directional pin it is connected to must be deactivated so that no driver contention is caused on the net. By keeping one of the bi-directional pins deactivated, the other pin sharing the same control cell remains deactivated at all times as well. Thus, the two bi-directional pins can be used as inputs only, which results in a reduced diagnosability. Opens, shorts and stuck-at faults can be detected on both nets, but opens cannot be localized.

The discussion above presumed that there are no active non-Boundary Scan pins included in the nets shown in figure 2. In reality, that is usually not the case though. Rather, pins from passive components, such as resistors, capacitors, connectors, etc., or analog, mixed-signal, and digital components may be connected to the same net. To safely test such nets using Boundary Scan the ATPG tools need an understanding of these pin functionalities and, specifically, if and how active non-Boundary Scan pins can be deactivated [7]. Ideally, non-Boundary Scan pin functionality is described in device libraries, so that it is easily available for automated test generation. If the ATPG tools are able to analyze the pin functionality, that information can be used to automatically generate “safe test vectors” – test pattern include automatically set constraints ensuring the avoidance of bus contentions in case there are no faults on the interconnections. Analyzing the non-Boundary Scan pin functionality also allows the ATPG tool to generate test programs with the best possible coverage at the smallest possible number of test vectors.

In Figure 3, below, two bidirectional buffers and a memory device share the same data bus. Any Boundary Scan tests need to be generated so that only one of these devices drives the bus signals at any time (“safe test”). In this example, the direction of the two buffers should be set so that they don’t drive against each other and the memory device should be disabled. Both buffer devices should be activated; this way the signal path from one Boundary Scan component to the other can be tested as part of the interconnect test and open faults on the buffer devices and shorts on the bus structure shared by the buffers and the memory can be detected. Alternatively, the two buffers could be disabled during interconnect test and the circuitry could be tested as part of a cluster test. Open faults on the memory pins would be tested as part of a memory cluster test.

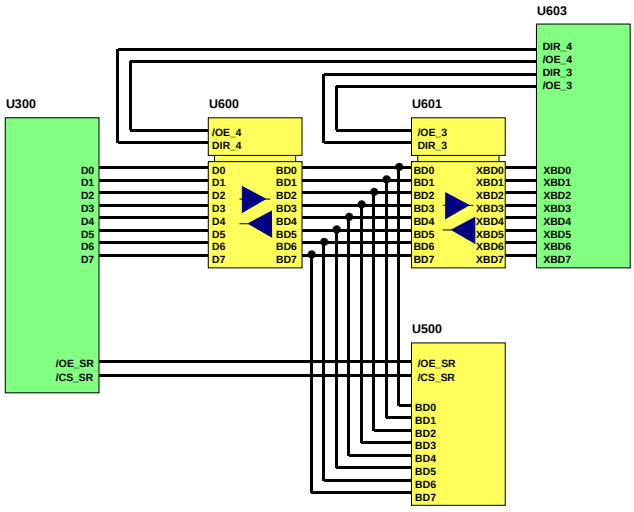


Figure 3: Bus structure

2.2 System level interconnect test

Board level test is often followed by system assembly and test. Main boards and daughter cards may be joined and modules may be plugged into a system backplane. It is beneficial to test such system level interconnects before attempting functional tests. Even when the system is dispatched in the field, remote and on-site system level test and In-System Configuration applications can reduce maintenance cost dramatically. Boundary Scan is a very

good candidate for such system level applications. To apply test pattern at a system level, the Boundary Scan infrastructure needs to be extended, though. Test vectors need to be routed from a controlling device (the Boundary Scan Controller) to individual devices or boards within the system. Scan paths may be split to group certain components in separate chains. Multidrop scan chain configurations for system level test require addressable chain routing devices [8].

A multidrop chain design at system level allows sharing one backplane IEEE-1149.1 test bus between all boards plugged into the backplane. A simple scan chain design, connecting all board scan chains in series at system level, would require all slots to be equipped; otherwise the scan chain would be broken. An addressable test bus interface on each individual card within the system provides for various levels of test, such as device level test, board level test, as well as system level interconnect test.

2.3 Limitations of IEEE-1149.1 interconnect test

A Boundary Scan Interconnect Test can only test connections (including transparent components) between Boundary Scan I/O pins. Such an interconnect test is running at relative slow speed and thus can be considered a static test. For example, with a Boundary Scan chain that is 10,000 cells long and a TCK frequency of 10MHz, the parallel test pattern throughput rate is approximately 1kHz (the rate at which the Boundary Scan I/O's can change their state during the test). That means that an IEEE-1149.1 interconnect test cannot detect connectivity problems on high speed interconnects that occur only at functional speed (e.g. cold solder joints that provide just enough connectivity for static tests, but cause functional faults at high speed because of a higher resistance).

Another limitation of Boundary Scan interconnect tests is that non-Boundary Scan pins (pins without IEEE 1149.x test resources) typically are not included in such tests. Exceptions are transparent devices such as buffers and transceivers, as well as serial resistors. To include other non-Boundary Scan pins in a Boundary Scan test, so called cluster tests would have to be executed.

3. Cluster Testing

Circuitry that is not directly testable via Boundary Scan we can consider a cluster. For example, combinatorial logic between Boundary Scan I/O pins can be called a logic cluster (see Figure 4). Power supply and distribution circuitry on a PCB is considered an analog cluster. Discrete (as opposed to embedded) memory devices, with or without external glue logic, are considered Memory Clusters. So called Interface Clusters comprise circuitry that connects the board to the outside world or provides visual or audio interfaces, respectively.

Often times, the input and output signals for such clusters, which may involve one or more individual components, are connected to Boundary Scan I/O pins. Utilizing those Boundary Scan I/O pins to stimulate the cluster inputs and to observe the cluster outputs, it is often times possible to create a basic functional test for the cluster. Since Boundary Scan test vectors are applied at a fairly slow rate, depending on scan chain length and TCK frequency, such cluster tests are usually not running at the same speed as the circuitry would be running at in functional mode.

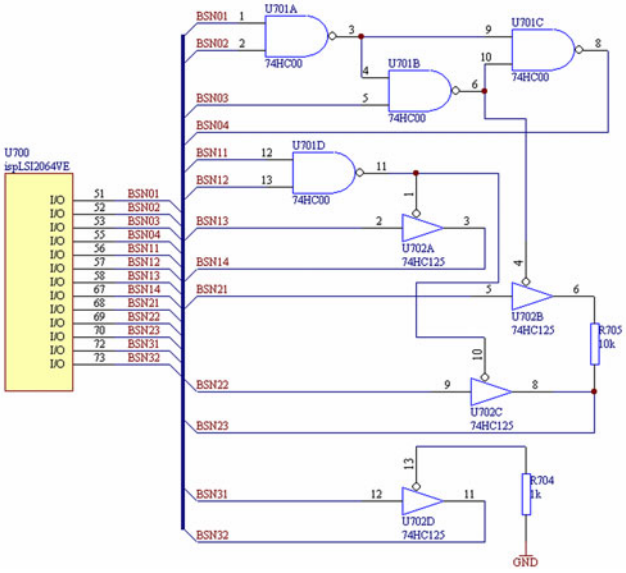


Figure 4: Example of a Logic Cluster

3.1 Logic and Interface Cluster testing

A Logic Cluster test verifies the connections from Boundary Scan pins to Cluster inputs and outputs and connections within the cluster as well as the general cluster functionality. Fault isolation is limited due to missing Boundary Scan access to all cluster-internal nodes.

In cluster testing it is as important to avoid bus contention as it is in any other kind of Boundary Scan test. Figure 4 shows a logic cluster where the cluster inputs and outputs are connected to the same Boundary Scan device. The logic cluster includes four 2-input NAND gates and four non-inverting signal buffers. Two of those buffers drive the same net. To avoid bus contentions, possibly resulting in damage of one of the drivers, only one of these two buffers may be enabled at any time. When creating test pattern for this circuitry either the developer or the automated test generation tool must find such nets that include multiple drivers and ensure that only one driver is active at any time.

An Interface Cluster test attempts to verify the interface connectivity and basic functionality. Since interface circuitry typically includes peripheral connectors, it is desirable to provide external test resources to those connectors to improve the testability and include the connectors in the test coverage.

Cases exist where generating cluster test with ATPG[9] tools is either not possible or not efficient. Here a high-level programming language is helpful, allowing the design engineer or the test engineer to manually write test source code at an abstract level while the Boundary Scan software tools handle the test resources at a low level (e.g. control of the TAP state machines, mapping of the Boundary Scan cells to be set, etc.). Such a programming language that allows the design or test engineer to concentrate on the test task itself, rather than on the underlying Boundary Scan protocols, can reduce test development time dramatically. Combining the flexibility a programming language provides with the insight and control graphical debugging tools can provide, Boundary Scan applications can be taken to the next level, up from just automatically generated interconnect and cluster tests to extended Boundary Scan applications that may even turn into integrated Boundary Scan and Functional Tests.

For an example, including test program sequence and source code snippets, please refer to Appendix A.

3.2 Memory Cluster testing

The purpose of a memory cluster test at board level is to determine that all memory pins are soldered properly, the memory is functioning, and there are no shorts, open, or stuck-at faults along the signal path from the controlling device – the Boundary Scan component(s) used to apply the test pattern – to the memory device. The memory cluster may or may not include glue logic and/or buffer circuitry. Figure 5 shows an example for a SRAM[10] memory cluster that includes buffer components on the address bus and data bus.

To test the connections between the Boundary Scan components (U300, and U603 in Figure 5) and the SRAM (U500 in Figure 5), a sequence of test pattern is written to the memory. When the respective memory locations are read afterwards, the same pattern should be returned by the memory. The test pattern must include control for the buffer devices also (U600, U601, and U604 in Figure 5). In the example in Figure 5 the memory device has 18 address lines and 16 data lines.

Memory Cluster tests usually are generated by ATPG tools. However, a functional description for the memory has to be provided to the ATPG tool. To simplify test development and reduce the development time, functional device descriptions (see Appendix B for an example) can be provided in device libraries. Boundary Scan software can rely on models to provide the necessary information for write and read access as well as memory setup sequences. Similar, functional descriptions for buffer and transceiver allow the software to automatically generate control pattern for such devices. All the ATPG tool has to do then is to match up the memory pins with the proper Boundary Scan I/O pins to apply the test pattern and to generate the test pattern itself. Latter typically depends on the number of address and data lines as well as on the type of memory. The goal is not to test the complete memory storage space for proper functioning, but rather to determine the structural integrity of the connections between the controlling Boundary Scan device and the memory. Here, as with any other Boundary Scan test application, it is important that the ATPG tool generates safe test vectors, that don't cause harm to the Unit Under Test.

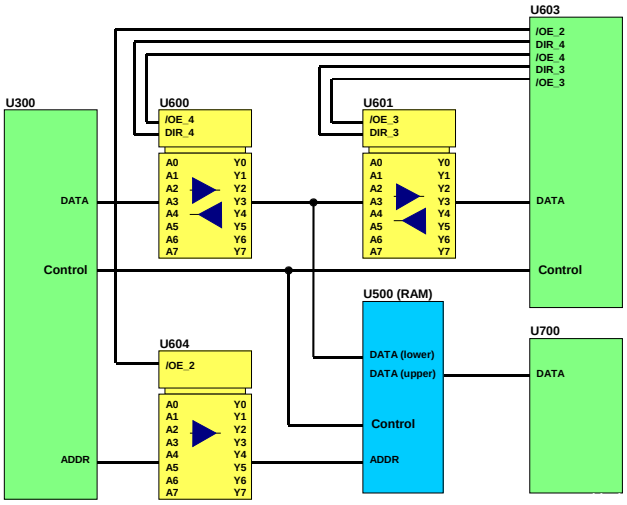


Figure 5: Example of a Memory Cluster

To test for opens and shorts, a test sequence such as the one presented in Figure 6 can be applied.

For an example of a SRAM memory cluster test sequence see Appendix C.

For a standard asynchronous SRAM device the write and read cycles are very simple; only three control signals are required (Chip Enable, Write Enable, and Output Enable). In addition to the control signals on the memory, the buffer devices need to be controlled to enable access to the SRAM from the Boundary Scan component (see Figure 5). The buffer devices themselves become part of the memory cluster; if the Boundary Scan software detects a fault on an address or data line, the fault location can be on the Boundary Scan pin, one of the buffer pins (input or output), or the SRAM pin. Of course, there could even be multiple faults on this signal path. A fault

[9] ATPG: Automated Test Pattern Generation
[10] SRAM: Static Random Access Memory

on the signal path can be detected, but if the connections between the Boundary Scan device and the memory device are not direct but rather go through buffers or glue logic, diagnosability is limited. Only if there are several Boundary Scan devices available on the address and data bus, the Boundary Scan pins themselves may already be verified as part of the Interconnect Test.

The example given here represents a very simple memory cluster. Developments in memory architecture in the past decade introduced a wide variety of new types of memories with often times rather complex control cycles required to access a memory device (to write and read data). In the following paragraphs we will discuss some of the most common memory types.

3.2.1 Types of memory devices

Memory devices can be classified as either volatile (e.g. SRAM, DRAM[11], FIFO[12]) or non-volatile memory (e.g. EPROM, FLASH EEPROM).

RAM (Random Access Memory) can be classified as static or dynamic memory. A Static RAM (SRAM) cell stores data in a flip-flop. The device retains the memory as long as power is supplied (hence the term static). A Dynamic RAM (DRAM) stores data as an electrical charge, which gradually discharges, and thus requires periodical refresh/ access to retain its data (hence the term dynamic).

Memories are also differentiated by the way they are accessed: synchronous vs. asynchronous. An asynchronous memory does not depend on a clock to write or read data, whereas synchronous memories are synchronized with an external clock signal (with the rising and falling clock edges, to be specific).

SRAM devices, for example, are available as both asynchronous and synchronous memory. Synchronous SRAM devices (SSRAM) are offered in a wide variety of interface implementations. All have input registers which latch the address, data, and control signals at the rising edge of the clock signal. Some also feature an output register for data. Depending on the implementation, we differentiate between Single Data Rate, Double Data Rate, and other memory interfaces.

With Single Data Rate (SDR) SSRAM, one data word is transferred between memory and controller per clock cycle (on the rising clock edge). Such SDR SSRAM come in different flavors: Pipelined, Flowthrough, Burst, Network. Pipelined SDR SSRAM offer both input registers as well as an output register, latched at the rising clock edge. This means that a write access to a pipelined SDR SSRAM takes one clock cycle, while a read access takes two clock cycles. Flowthrough SDR SSRAM, on

[11] DRAM: Dynamic Random Access Memory
[12] FIFO: First-In/First-Out

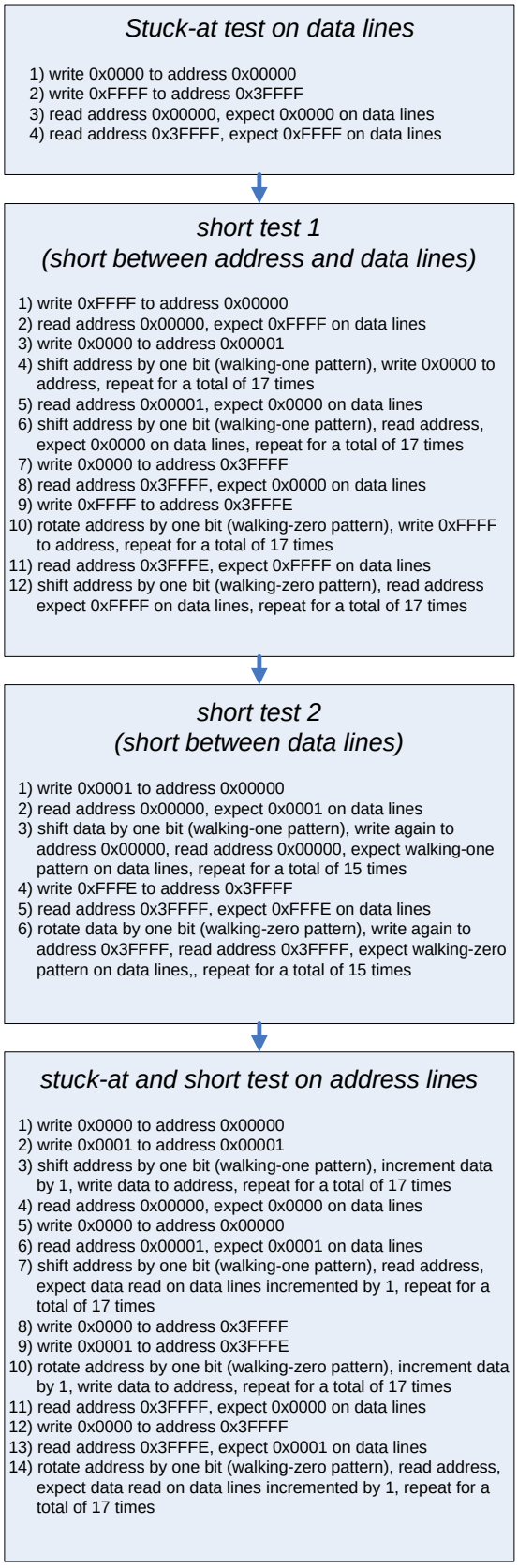


Figure 6: Example for RAM Cluster test pattern

the other hand, do not have an output register, thus both write and read access are one clock cycle long. Burst and Network SDR SSRAM provide built-in circuitry to enhance the data throughput, e.g. by reading multiple words from sequential addresses or by interleaving write and read accesses.

Double Data Rate (DDR) SSRAM can transfer 2 words per clock cycle, one on the rising edge and one on the falling edge of the clock signal. Thus, the data lines on such devices operate on double the clock frequency. A variant of DDR SSRAM, offering two independent ports for read and write access, is called QDR SSRAM (Quad Data Rate SSRAM) [9]. Each of the two ports on QDR SSRAM transfers data at a rate of 2 words per clock cycle, thus up to 2 words can be written to the memory and up to 2 words can be read from the memory within one clock cycle.

SDRAM memories are also available with various interface implementations (e.g. DDR-SDRAM, DDR2-SDRAM, RL-DRAM, RDRAM – a.k.a. Rambus – and others).

FIFO memory devices feature a variety of interface implementations as well, for example Clocked FIFO, Dual Port FIFO, and Quad Port FIFO, just to name a few.

Common to all of these memory interfaces is that they are much more complex than a standard SRAM interface. There are more than three control signals to be handled and often times a clock signal is involved. To be able to test the connectivity between a controlling device and the memory device within the scope of a memory cluster test utilizing IEEE-1149.1 resources, full Boundary Scan access to all required control signals as well as to the address and data lines is required [10]. However, especially the clock signal used to synchronize memory access cycles may not be accessible via Boundary Scan, but rather, it may be generated by an oscillator. If control over that clock signal cannot be obtained, the memory cluster test cannot be executed successfully.

3.2.2 Limitations on testability due to glue logic and clock circuitry

As long as all pins on a memory device are connected directly or indirectly (e.g. through buffer) to Boundary Scan I/O pins, a cluster test can be created for the memory. However, if the clock signal on a synchronous memory device (e.g. a SDRAM) is not controllable via Boundary Scan, the test pattern on address and data lines as well as the memory control pins cannot be synchronized to the clock and a test cannot be executed successfully. To take control over a clock signal coming from an oscillator for example, one could add a gating circuitry that is rerouting the memory clock signal to a Boundary Scan I/O pin whenever the cluster test is running. Figure 7 illustrates an example for such a clock gating circuit.

Here, the oscillator is enabled through the pull down resistor Roe. By providing additional control on the oscillator's /OE input pin (e.g from an otherwise unused Boundary Scan I/O pin on a PLD/FPGA, or from an external tester resource), the oscillator can be deactivated and an alternative clock source can be provided. In above diagram, an AND gate is used to gate the two clock sources (on-board oscillator and alternative clock source). While one of the two clock sources is active, the other input of the AND gate is kept at logic High via a pull-up resistor.

The more complex a memory interface is, the more Boundary Scan shift cycles are needed to write one data word to the memory or to read a data word from the memory. This results in increased test execution time.

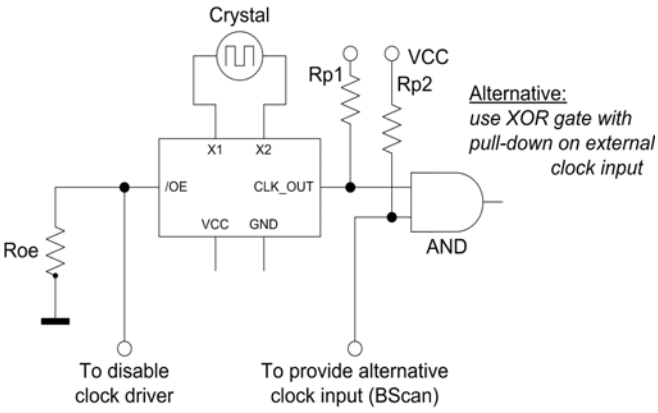


Figure 7: Clock gating to obtain BScan control

In addition to taking control over the clock signal for synchronous memories, a memory cluster test can only be executed successfully, if any glue logic that may exist to decode bank select, chip select, and other memory control signals is Boundary Scan accessible. The same is true for any bus switching circuitry that may be used to multiplex address and data lines. As mentioned before, the controlling Boundary Scan component(s) needs full access to the memory device (directly or indirectly) to be able to emulate write and read access cycles.

We mentioned earlier in this paper that dynamic RAM require periodical refresh of the data, otherwise the memory contents will be lost. The required refresh rate is so high that explicit Boundary Scan access cycles are too slow to refresh the data and at the same time run a memory cluster test. Typically this is not a problem though, since most modern dynamic RAM devices feature a self refresh or auto refresh algorithm (before applying the test pattern, the memory device can be set up to refresh itself periodically).

3.3 In-System Configuration applications

3.3.1 PLD and FPGA

Most modern programmable logic devices (PLD) and FPGA devices provide an IEEE-1149.1 test bus interface and support in-system configuration through that port. Typically, programming control is built in to the device and the control sequence and programming data is provided by means of various file types, such as SVF files, JAM files, STAPL (JEDEC-Std. 71) files [11], or IEEE 1532 files [12]. (As a note regarding location of PLD in the scan chain: If a PLD blocks data from scanning out of its TDO during in-system configuration, keep this device in a separate scan chain or put it at the end of the scan chain, with no other BScan components between this device and TDO of the board, or provide the means to temporarily isolate the device for ISP.) Note that FPGA devices typically feature configuration control pins that have to be externally driven to a certain logic level to make the component IEEE-1149.1 compliant. The BSDL file and data sheet for the component provides respective information. Modern FPGA often feature programmability of I/O pin functionality and voltage level. Thus, one has to decide whether to program the component before test or run Boundary Scan tests first (before device configuration).

3.3.2 EEPROM

FLASH devices and other EEPROM (such as serial EEPROM based on I2C or SPI protocol) can be programmed via Boundary Scan devices if access is available to all memory pins required for programming (either directly or indirectly). To reduce programming time, a short scan chain and high TCK frequency are required. Separate the BScan device used to program EEPROM from other BScan devices (put it in a separate scan chain) if those other devices support only a much slower TCK frequency. Also, try to control all EEPROM pins from the same BScan component (so that all other BScan components can be kept in HIGHZ, CLAMP, or BYPASS mode). Finally, programming speed can be increased if frequently exercised control pins (such as /WE) are accessed with parallel I/O resources rather than Boundary Scan. Precondition for that is that the Boundary Scan pin in that net can be disabled and that access to the control pin is available via connector (preferably) or test pad. Other techniques to speed up FLASH programming via Boundary Scan are in development or are already available as proprietary implementations.

Testing connection to a FLASH device by means of a memory cluster test such as described in 3.2 above would require multiple FLASH erase cycles, which typically is impractical because of the length of time that would be required for such a test. From a test engineer's standpoint it would be desirable to have IEEE-P1581 (see section 4.3) implemented in FLASH devices for connectivity test. Even better would be an implementation of IEEE-1149.1 and IEEE-1532 resources.

4. Alternatives to Memory Cluster test

There are alternatives to memory cluster tests based on Boundary Scan I/O pins surrounding the cluster.

4.1. Memories with IEEE 1149.1 test resources

Some vendors now offer memory devices with IEEE-1149.1 resources built in. Sometimes, these devices are not fully compliant to the IEEE 1149.1 standard for example by not supporting EXTEST capability. Such devices typically do support a test mode where the output drivers are deactivated and all I/O pins (address, data, and most control signals) provide capture capability. This – even though limited – test capability allows at least to include such memory devices in an automatically generated interconnect test.

Examples for memory devices featuring IEEE 1149.1 test resources include DDR, QDR, RLDRAM, and FIFO memory from vendors such as CYPRESS, GSI Technology, IDT, INFINEON, ISSI, MICRON, NEC, SAMSUNG, SONY, and others. [13,14]

4.2. Memory BIST

Memory embedded into Processors, SOC's[13], ASIC's[14], DSP's[15], and other devices often times can be tested with Built-In Self Test (BIST) resources implemented into these devices. In case of discrete memory at board or system level, a connectivity test between a memory device and its controlling counterpart (e.g. a processor) could be run at-speed if the controlling device has test resources embedded that can be used to apply at-speed test vectors that exercise the memory circuitry.

4.3. IEEE P1581

Currently in the balloting process, this standard [15] has been developed to define a test strategy for complex memory devices which do not support IEEE 1149.1. This standard describes a means to verify the memory I/O pin connectivity (address, data, and control signals). The memory cell structure is completely bypassed; a combinational test circuitry implemented into the memory device is used instead during test mode to link input and output signals. A controlling device (typically an IEEE 1149.1 compliant component connected to the memory device) applies a stimulus to the inputs and observes the outputs of the memory's test circuitry.

---

[13] SOC: System On Chip
[14] ASIC: Application Specific Integrated Circuit
[15] DSP: Digital Signal Processor

5. Outlook for Boundary Scan

The authors of this paper expect that in the coming years even more complex memories will feature IEEE 1149.1 or IEEE 1581 test resources.

For advanced applications of Boundary Scan, IEEE 1149.4 and IEEE 1149.6 will most likely become available in ASIC components first, since these types of Boundary Scan require a more sophisticated design of the test resources.

Furthermore, BIST for both embedded logic and memory circuitry based on IEEE P1500 is likely to become established quickly for SOC level testing once the standard is approved.

Extending the reach of Boundary Scan, integrations with other test methodologies such as Flying Probe, Automated Optical Inspection, and Functional Test improve cluster testing and achievable fault coverage.

6. Conclusions

Today, IEEE 1149.1 is well established and widely used. Still, there are many areas on even the latest board designs that require cluster testing because devices have no Boundary Scan test resources implemented. Memory devices are continuously becoming more complex and provide faster speeds and more storage capacity [16]. The bad news for test engineers is that logic cluster and memory cluster testing becomes more complicated or impossible on more complex UUT's. Design For Testability is more important than ever. The good news is that SOC's include more and more previously discrete circuitry, thus removing clusters from the board level, embedding it into the device; as a result BIST becomes more important.

7. References

[1] IEEE Computer Society, *IEEE Standard Test Access Port and Boundary Scan Architecture - IEEE Std. 1149.1 2001*, IEEE, New York, NY, 2001

[2] Kenneth P. Parker, *The Boundary-Scan Handbook, 3rd Edition*, 2003, Kluwer Academic Publishers, Norwell, MA, 02061, ISBN: 1-4020-7496-4

[3] IEEE Computer Society, *IEEE Standard for a Mixed-Signal Test Bus - IEEE Std. 1149.4 1999*, IEEE, New York, NY, 1999

[4] Adam Osseiran, *Analog and Mixed-Signal Boundary-Scan, A Guide to the IEEE 149.4 Test Standard*, 1999, Kluwer Academic Publishers, Norwell, MA, 02061, ISBN: 0-7923-8686-8

[5] IEEE Computer Society, *IEEE Standard for Boundary Scan Testing of Advanced Digital Networks - IEEE Std. 1149.6 2003*, IEEE, New York, NY, 2003

[6] Bill Eklow, Carl Barnhart, Mike Ricchetti, Terry Borroz, *IEEE 1149.6 – A Practical Perspective*, Proceedings of International Test Conference 2003, Paper 19.1, pp.494-502

[7] Bill Eklow, Richard Sedmark, Dan Singletary, and Toai Vo, *Unsafe Board States During PC-Based Boundary Scan Testing*, Proceedings of International Test Conference 2001, Paper 22.3, pp.615-623

[8] Clayton Gibbs, "Backplane Test Applications For IEEE Std. 1149.1", Proceedings of International Test Conference 2003, Paper 43.1, pp 167-180

[9] QDR Development Team website, http://www.qdrsram.com/

[10] Heiko Ehrenberg, *White Paper: Design-For-Testability Guidelines for Boundary Scan Test*, GOEPEL Electronics, 2004

[11] JEDEC, *EIA/JEDEC JESD71 – Standard Test And Programming Language (STAPL)*, EIA, Arlington, VA, 1999

[12] IEEE Computer Society, *IEEE Standard for In-System Configuration of Programmable Devices - IEEE Std. 1532 2002*, IEEE, New York, NY, 2002

[13] IDT datasheet for 3.3V High-Density Supersync II™ Narrow Bus FIFO IDT72V2113

[14] MICRON datasheet for RLDRAM II MT49H8M36, featuring IEEE 1149.1

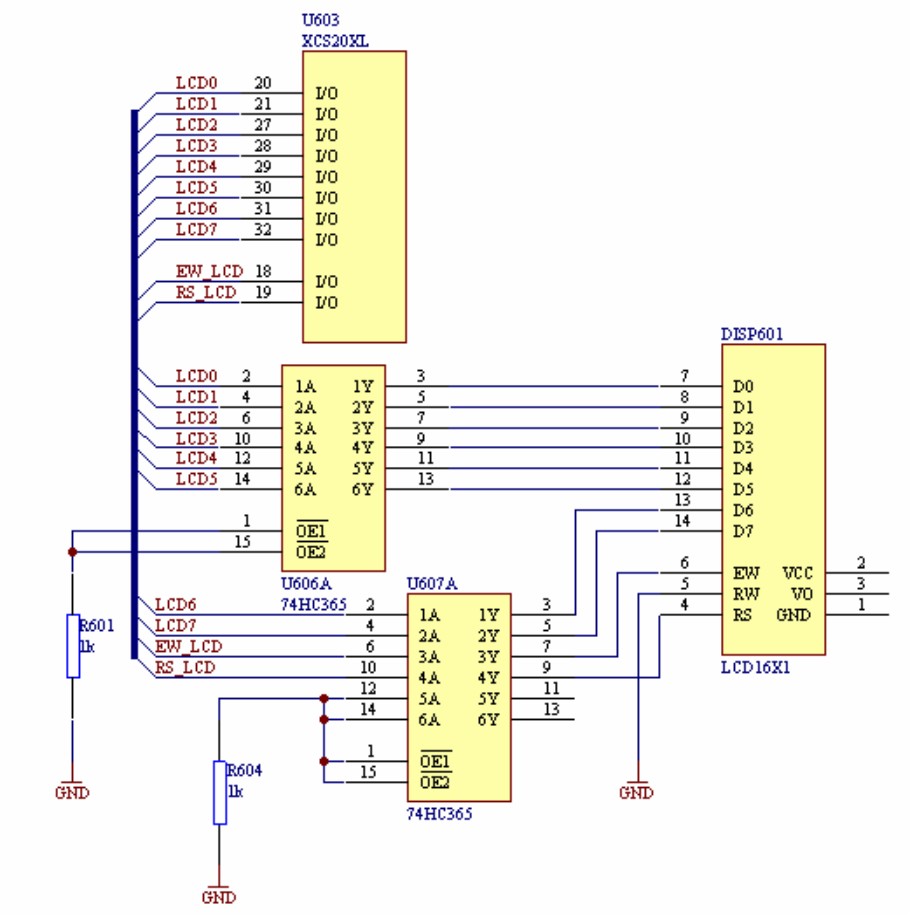
[15] IEEE P1581 website http://grouper.ieee.org/groups/1581/

[16] www.MemoryStrategies.com

Appendix

A. Logic Cluster test example

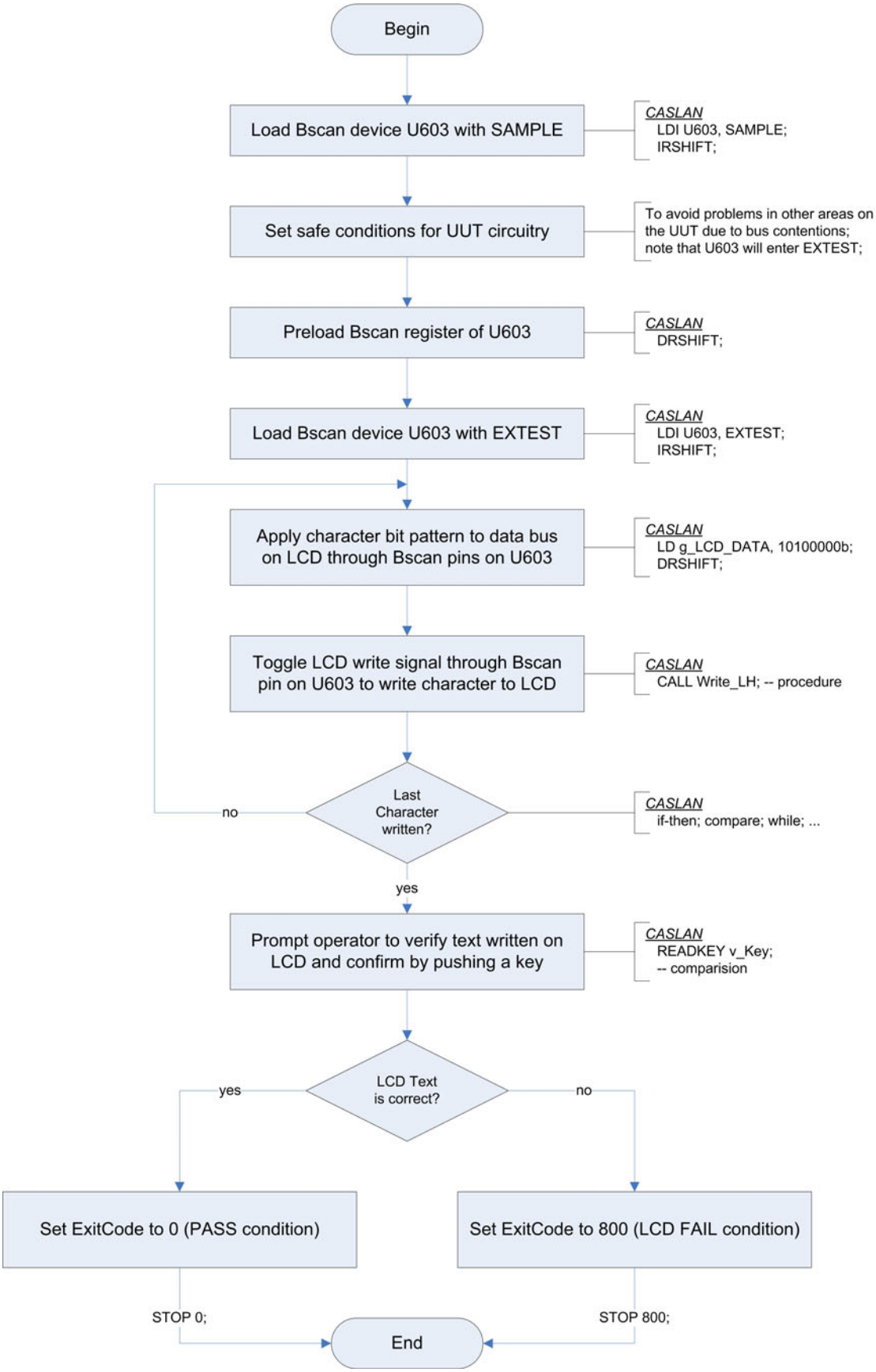
LCD display cluster



The advantage of Boundary Scan tools that feature a high-level programming language in addition to ATPG tools is that they provide a much better flexibility and support test applications that could not be covered with just automatically generated test programs. A programming language allows for the development of test programs with branches and conditional loops, vs. linear test pattern generated with ATPG tools. Examples for applications where a programming language is very beneficial include sequential logic and circuitry for user interfaces (such as LED and LCD displays, keyboards, speakers, etc.), where a Boundary Scan test of such circuitry borders on functional test applications.

A test for the LCD display cluster pictured above can easily be created manually in a high-level programming language. If combined with Automated Optical Inspection tools, the test pattern written out to the LCD could automatically be visually verified. Below is an example of a test flow for such an LCD cluster, with manually created test program snippets in the CASLAN[16] language.

[16] CASLAN: CASCON Language, proprietary Boundary Scan programming language by GOEPEL electronic


Begin
Load Bscan device U603 with SAMPLE
CASLAN
LDI U603, SAMPLE;
IRSHIFT;
Set safe conditions for UUT circuitry
To avoid problems in other areas on the UUT due to bus contentions; note that U603 will enter EXTEST;
Preload Bscan register of U603
CASLAN
DRSHIFT;
Load Bscan device U603 with EXTEST
CASLAN
LDI U603, EXTEST;
IRSHIFT;
Apply character bit pattern to data bus on LCD through Bscan pins on U603
CASLAN
LD g_LCD_DATA, 10100000b;
DRSHIFT;
Toggle LCD write signal through Bscan pin on U603 to write character to LCD
CASLAN
CALL Write_LH; -- procedure
Last Character written?
no
yes
CASLAN
if-then; compare; while; ...
Prompt operator to verify text written on LCD and confirm by pushing a key
CASLAN
READKEY v_Key;
-- comparision
LCD Text is correct?
yes
no
Set ExitCode to 0 (PASS condition)
Set ExitCode to 800 (LCD FAIL condition)
STOP 0;
STOP 800;
End

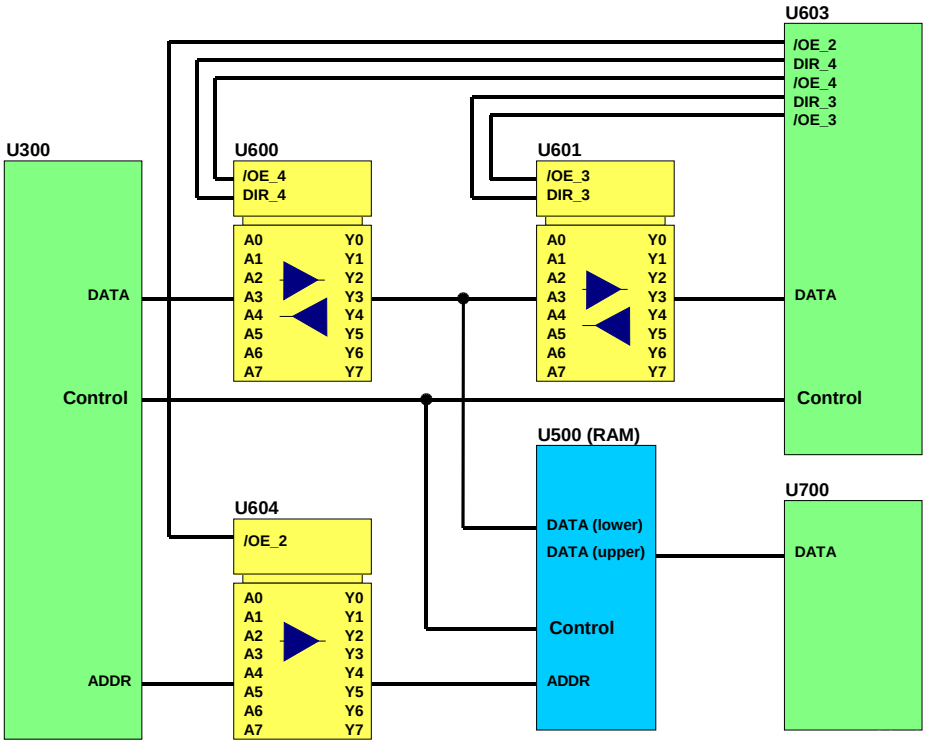
B. Device model description for a SRAM device

```
Format.............: BSDM
Version............: 4.0
Program............: 'CASCON DEVICE LIBRARY'
Remark......:
"
SRAM 256KX16
KM6164000_TSOP44 (SAM)


19-04-99
TK / GOEPEL ELECTRONIC
BR
V
"

Component name...................: KM6164000_TSOP44 (NonBSC)
Type.............: 'Ram'
-- Port table
-- Name Type
Port..: A4                  'Normal'     'Input'
Port..: A3                  'Normal'     'Input'
Port..: A2                  'Normal'     'Input'
Port..: A1                  'Normal'     'Input'
Port..: A0                  'Normal'     'Input'
Port..: CS                  'Normal'     'Input'
Port..: IO1                 'Normal'     'Bidirectional' 'Output2'
Port..: IO2                 'Normal'     'Bidirectional' 'Output2'
Port..: IO3                 'Normal'     'Bidirectional' 'Output2'
Port..: IO4                 'Normal'     'Bidirectional' 'Output2'
Port..: VCC1                'Normal'     'Linkage'       'SUPPLY'
Port..: VSS1                'Normal'     'Linkage'       'GROUND'
Port..: IO5                 'Normal'     'Bidirectional' 'Output2'
Port..: IO6                 'Normal'     'Bidirectional' 'Output2'
Port..: IO7                 'Normal'     'Bidirectional' 'Output2'
Port..: IO8                 'Normal'     'Bidirectional' 'Output2'
Port..: WE                  'Normal'     'Input'
Port..: A17                 'Normal'     'Input'
Port..: A16                 'Normal'     'Input'
Port..: A15                 'Normal'     'Input'
Port..: A14                 'Normal'     'Input'
Port..: A13                 'Normal'     'Input'
Port..: A12                 'Normal'     'Input'
Port..: A11                 'Normal'     'Input'
Port..: A10                 'Normal'     'Input'
Port..: A9                  'Normal'     'Input'
Port..: A8                  'Normal'     'Input'
Port..: NC2                 'Normal'     'Unknown'
Port..: IO9                 'Normal'     'Bidirectional' 'Output2'
Port..: IO10                'Normal'     'Bidirectional' 'Output2'
Port..: IO11                'Normal'     'Bidirectional' 'Output2'
Port..: IO12                'Normal'     'Bidirectional' 'Output2'
Port..: VCC2                'Normal'     'Linkage'       'SUPPLY'
Port..: VSS2                'Normal'     'Linkage'       'GROUND'
Port..: IO13                'Normal'     'Bidirectional' 'Output2'
Port..: IO14                'Normal'     'Bidirectional' 'Output2'
Port..: IO15                'Normal'     'Bidirectional' 'Output2'
Port..: IO16                'Normal'     'Bidirectional' 'Output2'
Port..: LB                  'Normal'     'Input'
Port..: UB                  'Normal'     'Input'
Port..: OE                  'Normal'     'Input'
Port..: A7                  'Normal'     'Input'
Port..: A6                  'Normal'     'Input'
Port..: A5                  'Normal'     'Input'
Package.............:PackageBegin 'Package'

Pins..:Packagebegin
  A4                  1,
  A3                  2,
  A2                  3,
  A1                  4,
  A0                  5,
```

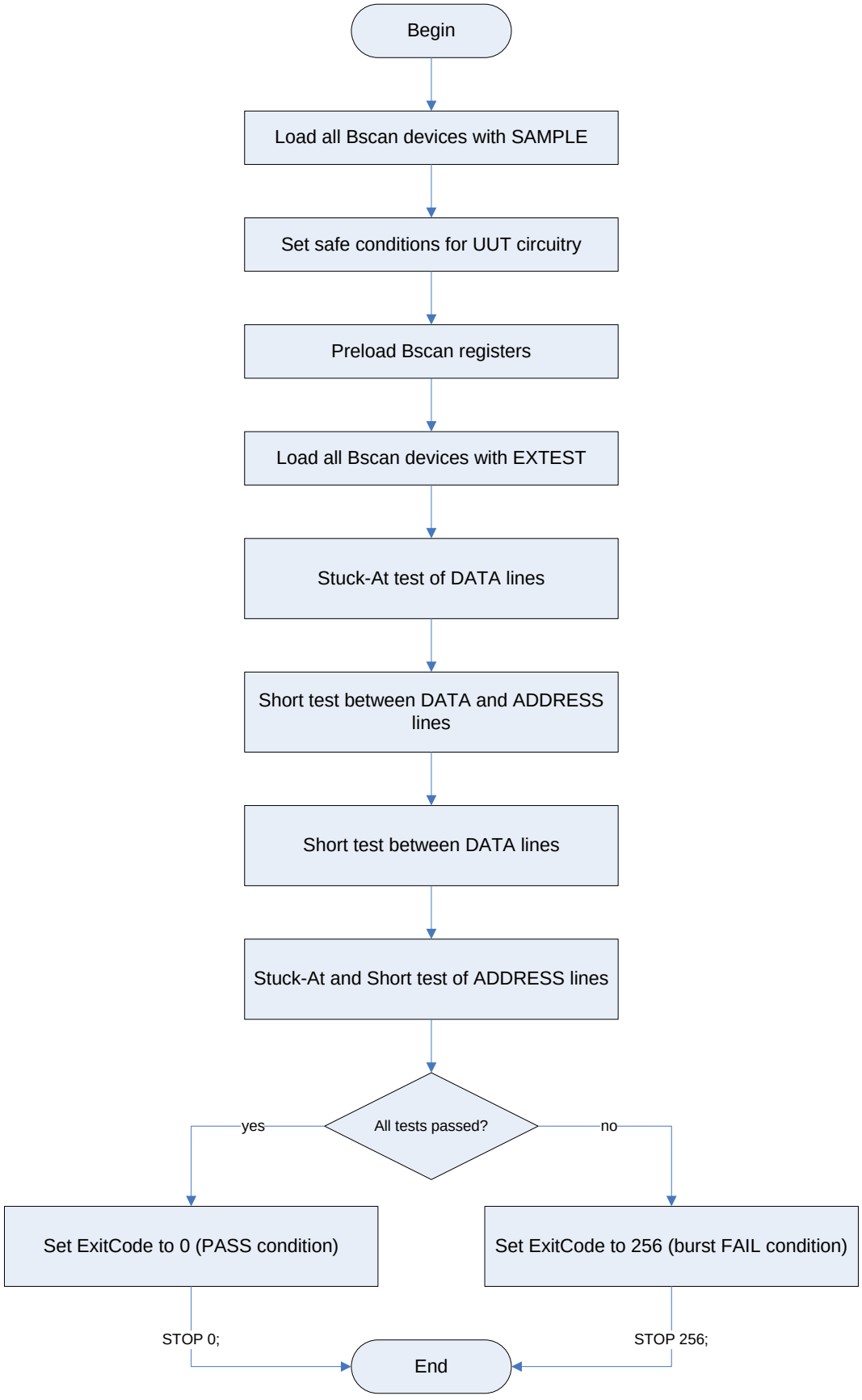
```
  CS                  6,
  IO1                 7,
  IO2                 8,
  IO3                 9,
  IO4                 10,
  VCC1                11,
  VSS1                12,
  IO5                 13,
  IO6                 14,
  IO7                 15,
  IO8                 16,
  WE                  17,
  A17                 18,
  A16                 19,
  A15                 20,
  A14                 21,
  A13                 22,
  A12                 23,
  A11                 24,
  A10                 25,
  A9                  26,
  A8                  27,
  NC2                 28,
  IO9                 29,
  IO10                30,
  IO11                31,
  IO12                32,
  VCC2                33,
  VSS2                34,
  IO13                35,
  IO14                36,
  IO15                37,
  IO16                38,
  LB                  39,
  UB                  40,
  OE                  41,
  A7                  42,
  A6                  43,
  A5                  44
 Packageend

Description...:Descriptionbegin
(FUNCTION
 (BUS
  (ENABLECONDITION (LOW OE)
                   (LOW CS)
                   (LOW LB)
                   (HIGH WE)
  )
  (PIN IO1 := ACTIVE)
  (PIN IO2 := ACTIVE)
  (PIN IO3 := ACTIVE)
  (PIN IO4 := ACTIVE)
  (PIN IO5 := ACTIVE)
  (PIN IO6 := ACTIVE)
  (PIN IO7 := ACTIVE)
  (PIN IO8 := ACTIVE)
 )
 (BUS
  (ENABLECONDITION (LOW OE)
                   (LOW CS)
                   (LOW UB)
                   (HIGH WE)
  )
  (PIN IO9 := ACTIVE)
  (PIN IO10 := ACTIVE)
  (PIN IO11 := ACTIVE)
  (PIN IO12 := ACTIVE)
  (PIN IO13 := ACTIVE)
  (PIN IO14 := ACTIVE)
  (PIN IO15 := ACTIVE)
  (PIN IO16 := ACTIVE)
 )
 (RAM
   (Address
```

```
    (Pin A0)
    (Pin A1)
    (Pin A2)
    (Pin A3)
    (Pin A4)
    (Pin A5)
    (Pin A6)
    (Pin A7)
    (Pin A8)
    (Pin A9)
    (Pin A10)
    (Pin A11)
    (Pin A12)
    (Pin A13)
    (Pin A14)
    (Pin A15)
    (Pin A16)
    (Pin A17)
  )
  (Data
    (PIN IO1)
    (PIN IO2)
    (PIN IO3)
    (PIN IO4)
    (PIN IO5)
    (PIN IO6)
    (PIN IO7)
    (PIN IO8)
    (PIN IO9)
    (PIN IO10)
    (PIN IO11)
    (PIN IO12)
    (PIN IO13)
    (PIN IO14)
    (PIN IO15)
    (PIN IO16)
  )
  (Control
     (PIN CS)
     (PIN LB)
     (PIN UB)
     (PIN WE)
     (PIN OE)
  )
  (Inactive
    D, Z, HHHHH
  )
  (Write
    (Step D, D, LLLHH)
    (Step D, D, LLLLH)
    (Step D, D, LLLHH)
  )
  (Read
    (Step D, D HHHHHHHHHHHHHHHH, LLLHH)
    (Step D, Z, LLLHH)
    (Step D, E, LLLHL)
    (Step D, Z, LLLHH)
  )
 )
) Descriptionend
```

C. Test flow for a SRAM Cluster test



Below is a test flow for an automatically generated memory cluster test for the SRAM memory cluster pictured in the diagram above. See also Figure 6 in the main part of this paper. The ATPG tool needs to generate safe test pattern, making sure that no bus contentions occur (neither within this memory cluster nor throughout the rest of the circuitry on the Unit Under Test).


Begin
Load all Bscan devices with SAMPLE
Set safe conditions for UUT circuitry
Preload Bscan registers
Load all Bscan devices with EXTEST
Stuck-At test of DATA lines
Short test between DATA and ADDRESS lines
Short test between DATA lines
Stuck-At and Short test of ADDRESS lines
All tests passed?
yes
no
Set ExitCode to 0 (PASS condition)
Set ExitCode to 256 (burst FAIL condition)
STOP 0;
STOP 256;
End