Remote Boundary-Scan System Test Control for the ATCA Standard

David Bäckström*, Gunnar Carlsson+, Erik Larsson*

Embedded Systems Laboratory*
Linköpings Universitet
Sweden

Digital Processing Platform+
Ericsson AB
Sweden

***Abstract* -- The backplane in a multi-board system has a limited wiring capability, which makes additional backplane Boundary-Scan wiring to link the boards highly costly. The problem is to access the Boundary-Scan tested boards with the Boundary-Scan controller at the central board. In this paper we propose an approach suitable for the Advanced Telecom Computing Architecture standard where we make use of the existing I^2C-bus and the Intelligent Platform Management Bus (IPMB) protocol for application of operational tests. We have defined a protocol with commands and responses as well as a test data format for storing test data on the boards to support the remote execution of Boundary-Scan tests. For validation of the proposed approach we have developed a demonstrator.**

I. Introduction[1]

A modern system often consists of several printed circuit boards (PCB) connected through a back-plane. In such a multi-board system there is usually one central control board responsible for the control of the rest of the boards in the system. The boards in a system are usually equipped with Boundary-Scan to ease testing [1], [2]. In operation and maintenance testing the boards must be accessed through the control board and a linkage is required. The backplane can for this purpose be extended with additional wires for Boundary-Scan.

There are several commercial solutions available to link a Boundary-Scan bus from the test controller on the central board to the local Boundary-Scan infrastructure through the backplane environment [3], [4]. However, these solutions require additional Boundary-Scan wiring in the backplane.

In addition to additional wiring in the already crowded backplane, there is also a need for a well defined way to control and manage downloading, storing and execution of the on-board test sets. Today, almost every vendor of automatic test equipment (ATE) has its own specific API/ command set and data format to transport and store onboard tests. The lack of a standardized command set is probably one of the most important reasons for the low deployment of embedded Boundary-Scan today. Also, this has led to unnecessary difficulties when designing system tests for multi-board system where boards or parts are often made by several different vendors.

In this paper we present an approach to limit the backplane wiring problem. Our approach is suitable to be included for systems based on The Advanced Telecom Computing Architecture (AdvancedTCA or ATCA) standard; an architectural multi-board platform for carrier-grade telecommunication applications [5]. We make use of the existing and well-known maintenance architecture in the backplane given in the ATCA; the I^2C-bus [6] and the Intelligent Platform Management Bus (IPMB) [7] protocol. For the problem of lacking a standardized command set, we present a well defined command set and an embedded data format. This will provide both the ATE manufactures with a uniform application interface and the system designers with an easy to implement data and control structure. The presented command set together with the data format is able to handle the following three levels of test scenarios:

- Embedded go/no-go test.
- Embedded diagnostic test.
- Remote diagnostics.

We have developed a demonstrator to validate the proposed scheme. We assume that the reader of this paper has some basic knowledge about Boundary-Scan.

The rest of the paper is organized as follows. Section II gives an overview of related work and Section III introduces the preliminaries and the assumed environment. The problem definition is in Section IV and the proposed approach is described in Section V. Section VI contains a discussion, Section VII presents the demonstration and the paper is concluded with conclusions in Section VIII.

[1.] The research is partially supported by the Strategic Integrated Electronic Systems Research (STRINGENT) programme.

0-7803-9039-3/$20.00 © 2005 IEEE

II. Related Work

This section gives an overview of previous work that has targeted embedded Boundary-Scan in a multi-board environment. There are several solutions available how to link Boundary-Scan control from a central test controller to locally distributed test controllers. They could be divided into three groups;

- *Boundary-Scan Architecture and Protocol*. Boundary-Scan is added and its protocol is used,
- *Boundary-Scan Architecture and Extended Protocol*. Boundary-Scan is added and the protocol is extended, for instance with an addressing capability.
- *Alternatives to Boundary Scan*. A completely different bus architecture is used to wrap and transport the Boundary-Scan data to the desired boards.

We will now show a few examples of each group, and discuss their pros and cons.

A. Boundary-Scan Architecture and Protocol

The backplane is extended with Boundary-Scan and the Boundary-Scan protocol is used. There are alternatives to connect the boards; the ring-architecture and the star-architecture. These are well-known but rarely used schemes in larger multi-board systems, and they do not require any additional implementation beside the standard Boundary-Scan requirements.

In the ring-architecture a complete Boundary-Scan chain is daisy-chained through the backplane and all boards in the system. The approach creates a potentially long and cumbersome scan-chain to use. Such a solution in a multi-drop system also runs into problem when boards are removed or added, since it requires some sort of jumpers/bridges when a card is removed or the chain will be broken.

The star-architecture is based directly on a pure Boundary-Scan bus. In the approach every board in the system gets a dedicated TMS line and can then be controlled separately. However, such an approach requires a larger amount of connection lines in the backplane that might be available (*i.e.* one additional line for each card).

The advantage of the ring and star architectures is that they do not require any additional components or new protocols beside the required of the Boundary-Scan specification. This makes them straight forward to implement, but in a larger multi-board system they are often to cumbersome to use.

B. Boundary-Scan Architecture and Extended Protocol

The backplane is extended with Boundary-Scan and the Boundary-Scan protocol is extended. The approach is the one used mostly in today's systems. We will describe two of the most well used designs, the Addressable Shadow Port by Texas Instrument (TI) and the ScanBridge by National Semiconductor (NSC).

Whetsel [8], [9] presented a scheme where Addressable Shadow Ports (ASPs) are used to gain access to specific boards or scan-chains in a system. In this scheme a new protocol layer is added on the Boundary-Scan bus and used to link the backplane bus to the local scan-chain on the board before the actual testing commences. This new protocol is active when the Boundary-Scan logic is in its run-test/idle state, test-logic-reset state, pause-dr state or pause-ir state. It uses the then disabled TDI and TDO lines to facilitate a select/acknowledge protocol. Texas Instrument supplies interface components [3] that support this ASP scheme.

A variant of the approach was first proposed by Bhavsar [10] and later developed further by National Semiconductor [4] into the ScanBridge scheme. In ScanBridge there is also an overhead protocol, but instead of using the Boundary-Scan bus in its idle states it shifts an address like an instruction using the shift-ir state on the TDI line. The addresses of the boards (*i.e.* the local scan-chains) are known to the interface units, and only the interface unit with the matching address is connected to the backplane bus (Level 1). The rest of the boards are disconnected. However, before any shifting of any test data to the target board can commence, a "Level 2"-protocol is used to further select among local scan-chains on the board.

Both the TI solution and the NSC solution are tightly linked to the 4-wire (optionally 5 wires) Boundary-Scan bus as a backplane interconnection (with their own specific modifications to the protocol). These additional wires, especially when an already implemented maintenance bus is available in ATCA, might be a deal-breaking requirement.

In addition to the extra wiring in the backplane these solutions do not provide any detection of errors that may occur during backplane transmission. The result could expose the components-under-test to corrupted test and control data and in the worst case even damage the components. Ke *et al.* [11] proposed a novel scheme to include error detection into a standard Boundary-Scan backplane bus, such is used in the ASP and ScanBridge designs. The IPMB protocol used in our solution does however already include error detection features.

C. Alternatives to Boundary-Scan

An alternative is to not make use of Boundary-Scan but instead make use of a totally different bus and protocol to transport test data in the backplane. The solution presented in this paper falls into this group, because we use

the IPMB bus to transport and control the Boundary-Scan tests.

The withdrawn IEEE standard 1149.5, the "Standard Module Test and Maintenance (MTM) Bus Protocol" [12], was designed to facilitate both testing and other maintenance functions in multi-board systems. The problem with the standard was that while it did specify the message transport interface on both sides, it did not specify an embedded test data format or a command set to run specific tests. This was left open to the users of the standard. It resulted in that the standard was never really adopted by the industry and is now abandoned.

Instead the Intelligent Platform Management Interface (IPMI) framework and the Intelligent Platform Management Bus (IPMB) bus was developed and is today increasingly gaining momentum through the ATCA system architecture. The IPMB bus is well suited for additional functions and only requires 2 wires in the backplane compared to MTM that requires 5 wires and ASP and ScanBridge that require 4 each (standard Boundary-Scan Bus architecture).

Whetsel [8] also presented in his paper an expanded version of his proposed ASP scheme called Commendable ASP (CASP). Using CASP enables remote test access and data transfer operations. The CASP protocol also includes a cyclic redundancy check (CRC) for error detection that the ASP protocol didn't include. He also suggests that CASP (or ASP) is possible to convey over a 2-wire serial bus instead of the 4-wire standard Boundary-Scan bus. The CASP scheme is however still bound to a particular bus protocol and architecture (2 or 4-wire) that is not available in today's system architectures like ATCA. Therefore it is essential to make the command set and the data format as platform independent as possible by separating it from the design of the bus architecture.

III. Environment

This section contains a brief introduction to the two key components of this project, the Boundary-Scan technique [2] and the Advanced Telecom Computing Architecture standard (ATCA) [5].

A. Boundary-Scan testing

Boundary-Scan testing was developed to be a method of accessing test nodes on PCB's that was difficult or impossible to reach by conventional physical methods (*e.g*. the bed of nails technique) and the original goal was to facilitate interconnection tests at board-level [2].

Today, the Boundary-Scan standard is well-accepted and it has evolved from its first purpose to facilitate interconnection tests on PCB's to include advanced features like in system configuration [13] and embedded Built-In Self-Test (BIST) of components. The proposed implementation in this paper will enable usage of some of these new features in a multi-board system.

B. The ATCA system environment

The systems, which this paper is mainly aimed at, are telecom and networking applications, like telephone and optical switches. These systems will in the future be increasingly based on the ATCA-standard that specifies the mechanical building practice and the backplane interfaces [5].

The ATCA architecture is implemented using a *shelf* (backplane) with several *slots* where control boards and application specific boards (*blades*) can be inserted. Often a master control board is used to control the operation of the application specific boards. The operational communication (control and user data planes) between the boards is mainly going through the *main backplane connection bus(es)*.

Specified in the ATCA-design is also a platform management system [7] (*i.e.* IPMI), for which the major role is power and thermal management or the "health" of the system by monitoring different type of sensors etc (see Figure 1). This usually includes management of functions like fan control, power control, temperature and voltage levels readings. IPMI uses a simple bus and protocol IPMB to communicate between the boards. In turn, IPMB is physically implemented using the I^2C Bus [6] (a serial 2-wire, widely used bus) as carrier.

The protocol used on the IPMB is a request/response protocol. When a request message is sent to a device on the IPMB, the receiving device must respond using a defined response message.

In the IPMI concept, the platform management functions are controlled from a Shelf Master (the SM unit, sometimes referred to as Shelf Management Controller - ShMC), typically placed on a system control board. At the application boards, Baseboard Management Controllers (BMC units, sometimes referred to as IPM Control-

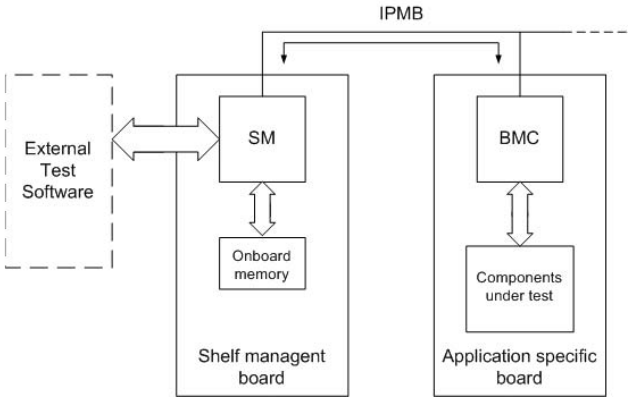

Figure 1: Basic IPMI structure.

ler - IPMC) are locally controlling the maintenance functions on the boards they sit on.

Note that IPMI is merely a framework for implementing new management functions. This paper suggests a way to implement the Boundary-Scan functionality into this structure.

IV. Problem definition

In this section we will clarify the problem which the proposed solution in this paper is intended to solve. We will do this by describing three test scenarios and their related requirements on data and control transport. In the end of this section we will summarize the scenarios into a few goals and overall requirements.

A. Embedded go/no-go test

Typical usage of embedded go/no-go test is at cold (re)start of a board in the field due to a severe alarm, or as a regular test at non-busy hours. Short test time is often crucial. Control of the test may be through a system maintenance (operator) interface or through intelligent maintenance SW.

In this kind of test the test-set is usually resident locally on the target board/module and only called to run by a simple command from the maintenance controller on the main control board. The analysis and comparison of the test response data is also done locally on the board. When completed, the board test controller responds with a small pass/fail message back to the maintenance controller on the main control board.

B. Embedded diagnostic test

Embedded diagnostic test is an extended test to detect potential HW faults that are not detected by the go/no-go test. The reason may be frequent alarms and restarts with suspected HW problems. A slightly longer test time may be allowed, if it results in a higher resolution of the test. Control of the test may be through a system maintenance (operator) interface.

This kind of test is used to gain more information from a given set of stored test vectors/programs. As before, the comparison is still made locally on the target board, but instead of only sending a small test report back to the controlling operator, the operator can select to retrieve a test log or even the actual response data if needed.

C. Remote diagnostics

Typical use of remote diagnostics is at diagnostic test in a reference system in the repair shop. The test will be more capable to pinpoint faults to components and interfaces. Control of the test may be through a system maintenance (operator) interface or from a connected external test system.

This is used when extensive testing and flexibility is needed. Beside the resident tests on the target board, new test sets can also be downloaded and run. The test responses is analyzed at the target or sent back to an external test system for comparison and further study. In this scenario longer test times are accepted due to the higher rate of availability and flexibility.

D. Goals and Overall Requirements

The requirements of these three scenarios together with the requirement that the solution should fit in the ATCA context and not violate the IPMI management standard present the goals for the suggested solution.

V. Proposed solution

This section will describe the proposed solution to the requirements stated in the previous section.

We will expand the usage of IPMI to include fault management, *i.e.* to let the IPMB carry Boundary-Scan test commands and data between the central SM unit and the locally distributed BMC units. This should be seen as a complement to the conventional way of having the main control processor controlling the testing of application boards, using the ordinary functional control path to the functional local board control processors. Also the function of IMPI/IPMB is separated from the normal operational functions in a system, and due to this, the IPMB could be regarded as a “system back door”, which allows testing even in case the ordinary control path is out of order. Using the IPMI/IPMB in this way does not violate the IPMI and the IPMB specifications.

The solution consists of four parts:

- A description of where and how the Boundary-Scan functionality is implemented in the ATCA/IPMI context.
- A specification of an API or set of commands used by a remote or an internal test program to manage the embedded tests and their execution.
- An embedded binary test vector format in which the locally onboard test sets will be stored. This is the actual instructions that a low-level Boundary-Scan HW-driver will execute.
- An extended IPMB protocol to support the transport of Boundary-Scan test and result files over the IPMB.

The test flow controller which receives and interprets the commands does require information about the HW-implementation/driver and similarly the low-level HW-driver does not require information about the structure of the command set. This segmentation of our solution into several layers offers many advantages. First it reflects the

typical different competence categories involved in the development of embedded tests. Secondly the separation between the test execution flow controls (*i.e.* the command set) and the low-level Boundary-Scan HW-driver enables reuse of the test controller in future applications of embedded Boundary-Scan test purposes. The second separation between the system communication links (and its management) and the test execution controls also eliminates the dependencies if commands are fetched from local onboard storage or sent from a remote site.

A. Added parts in IPMI

This subsection will describe the functionality of the new modules that have been added to the IPMI management system, and how they make use the already available functions.

The IPMI is merely a framework for implementing management functions (See Figure 1). It provides, among other things, interfaces and bus structures (*e.g.* IPMB) to support a distributed management system. The framework, with some additions, is well suited for implementing HW fault detection like the Boundary-Scan technique.

The main IPMI controller on the shelf management board (*i.e.* the SM unit) will also be the main controller when performing Boundary-Scan applications. It receives commands from the system manager interface and according to those gives commands to the local satellite controllers (*i.e.* BMC's) through IPMI interface. Most of the low-level test data will be stored on the local boards in the system, but some additional test sets might also be stored on or sent to the shelf management board under control of the SM unit. These additional test sets could be used when an extended embedded diagnostic is required on an application board (*i.e.* see the second test scenario above - Section IV.B). In this case the SM unit could even be performing some trivial comparison and analysis of the received test response. However, most of the time the SM unit will act like a bridge and command interpreter between the system management interface and IPMB link to the BMC's.

The local IPMI controllers on the boards are the units that perform the actual testing. They will receive commands and data from the SM unit through IPMB and act upon it. Figure 1 shows the BMC unit separated from the onboard memory, where the Boundary-Scan tests are stored, and from the components under test. The onboard test storage and the components under test are also separated so the coverage of the CUT are not reduced.

This is a short summary of the new functionality that has been added to the BMC:

- Write, read and manage onboard stored test sets.
- Run one or more onboard test sets.
- Perform comparison and straight forward analysis of the received test response.
- Logging of execution and results of test runs.
- Send test reports and logs to the SM unit when requested.

To ease some of the burden of the BMC implementation, especially when handling the serial interface of Boundary-Scan bus, a special *embedded Boundary-Scan controller* could be used. This kind of controller is basically just an asynchronous parallel to serial (Boundary-Scan) interface and has been commercial available for some time [14], [15].

B. Commands

This section contains a list and a description of each of the commands and the corresponding responses. The API enables the system management functions or a system operator an easy access to the onboard or remote embedded tests. The commands are divided into three groups:

- *embedded data management* commands,
- *test control* commands, and
- *support* commands.

1) Embedded Data Management Commands

The following *embedded data management* commands are available:

- LIST_TEST: Lists all test sets stored onboard in the BMC.
- LIST_LOG: Lists all test log entries stored onboard in the BMC.
- WRITE: Used to send/write and store new test sets in the BMC.
- READ: Used to retrieve/read stored test sets or logs in the BMC.
- DEL: Used to delete a specific test set or log entry in the BMC.

2) Test Control Commands

The following *test control* commands are available:

- SET_TEST_RESPONSE: Used to control if each test should return test response always, only on failure or never to the user.
- SET_TEST_MESSAGE: Used to control if each test should return its log entry to the user always, only on failure or never.
- SET_LOG: Used to set for each test set if the result should be logged or not.
- SET_STOP_CONDITION: Used to set if tests should abort on first failure or continue whatever happens.
- RUN: Used to run one or more selected test sets

stored in the BMC.

3) Support Commands

The last group of commands is the support commands:

- HW_OPTION: This command is used to give HW specific commands to the BMC, like setting the frequency of the TCK.
- HELP: Displays a list of available commands.
- EXIT: Stops all execution of the Boundary-Scan application and returns the Boundary-Scan logic to a non intrusive state.
- STATUS: Displays current option settings and system status.

These commands are given through a system management interface. A terminal emulator program capable of handling the X-modem protocol is required to send and receive files with the WRITE and READ commands .

To illustrate how these commands could be used in a typical system during more rigorous test/diagnostics (*i.e.* the test scenario in Remote diagnostic, Section IV.C). In the scenario the already stored test sets onboard is not enough to locate a possible HW-fault. To solve this we want to download a new test set to the BMC, run and log the test and retrieve the log and test response for further analysis on a connected workstation/pc.

1. First we use the STATUS command to check if the Boundary-Scan functionality itself is ok and current settings of the options.
   > STATUS
   System status: OK
   Data log: on_fail
   Stop condition: continue
2. Next we use the LIST commands to check what test sets are already stored onboard. We use the DEL commands to remove test set 2 from the memory to free some space for a new test set.
   > LIST_TEST

Set_num	Name	Log	Result
1	BIST	no	yes
2	logic_1	yes	yes

   > DEL test 2
   Test set 2 deleted.
3. Download the new test set using the WRITE command and the X-modem protocol. Give the file name as a parameter.
   > WRITE logic_2
   New test set received and stored at Set_num 2.
4. Run the downloaded test set with the RUN command. The parameter selects which test to run.
   > RUN 2
   Test completed, status: fail
   New log entry created for test 2.
5. List the log entries with the LIST_LOG command and retrieve the new log for further analysis at a PC with the REC command.
   > LIST_LOG

Log_num	Set_num
1	2

   > REC log 1
   Log_num 2 retrieved.

The example shows how the command set can typically be used to manage the new implemented modules in IPMI.

C. The Embedded data format

All Boundary-Scan test vectors, expected test response and control data are stored in the system in a format called Binary Vector Format (BVF) (Figure 2). It is essentially a compact binary version of the Serial Vector Format (SVF) [16] and all posts are all well defined to make it as memory organization and processor independent as possible.

All required operations (instructions and data) of a complete Boundary-Scan test set, like a full system BIST test, can be stored in one BVF-file. This results in that only one file needs to be downloaded to the system and stored on the onboard memory and thus reducing the memory overhead.

Every BVF-file is composed of a number of BVF partitions, and every BVF partition is composed of a number of BVF records. Every BVF partition starts with a header record and ends with an end of partition record (*i.e.* EOP). The rest of the records are a mix of these;

- BVF_SDR, Scan Data Record.
- BVF_SIR, Scan Instruction Record.
- BVF_RUN, Run Test Record.
- BVF_TRST, Test Reset Record.

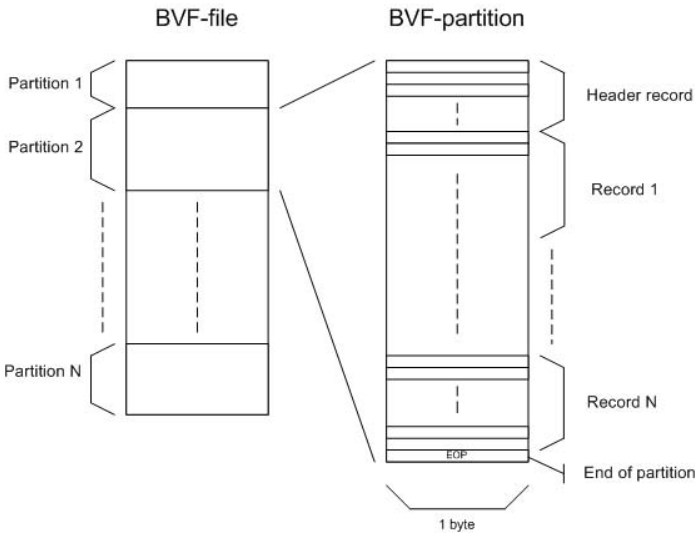

Figure 2: BVF-file format.

And these additional control records are also available:

- BVF_ENDDR, Default end state for DR operations.
- BVF_ENDIR, Default end state for IR operations.
- BVF_STATE, Forces the Boundary Scan logic to a specific state.

The header record and the end of partition record together with the operating and control records above are all well defined and start (*i.e.* the first byte in the record.) with an op-code. This op-code determines how the rest of the record will be structured.

The *header record* of each BVF-file consists of, beside the op-code, fields such as file size in bytes, test name and creation date.

The *end of a partition record* consists of one byte, which is its op-code that also identifies it.

The *scan records*, BVF_SDR and BVF_SIR, consists of their op-code, an option byte, number of bits to scan in, a field containing the data to be scanned in and an optional field containing the expected response data. The option field tells the Boundary-Scan executor, in our system the BMC unit, for example if comparison should be made onboard with the supplied expected test response or not.

The *run test record*, BVF_RUN, is used to wait in the Boundary-Scan logics Run-Test/Idle state for a number of cycles (*e.g.* when a BIST is running.). The record is short and simple; it consists of its op-code and four bytes with the number of cycles to remain in the Run-Test/Idle state.

The *test reset record*, BVF_TRST, consists of its op-code and an option field stating if the test logic should be put in a non intrusive reset state or not.

The three *control records*; BVF_ENDDR, BVF_ENDIR and BVF_STATE consists of their op-code and a byte containing one of the Boundary-Scan logic's stable states.

D. The Extended IPMB protocol

Originally, the IPMB was designed to carry short control and status messages within the IPMI management system. However, we also want to be able to transport larger Boundary-Scan test files and test results using the IPMB. For this purpose we extended the IPMB protocol. All the new extensions to the message structure are made within the data-field of the messages, so the extended messages are still valid to transport on any IPMB. The effect of this is that not all BMC units in a system need to be implemented with the ability to decode such messages, only those with the actual Boundary-Scan functionality. We have defined a new set of IPMB-commands for this purpose and assigned them to a special network function in the OEM-range to avoid conflicts with future specifications of the IPMI/IPMB.

In our approach the SM unit will always act as the requester, *i.e.* it will only send request messages, and the BMC units will only answer to this requests with response messages.

The maximum allowed messages size on the IPMB is limited to 32 bytes in the original specification. This forced us to implement a mechanism that partitions the Boundary-Scan test files into smaller packages that can be fitted inside a IPMB message. These smaller packages are then reassembled into a complete test file again on the receiver side.

We will not describe the details about the message structure of the requests and responses that are send between the SM and BMC units. However, the basic concept is that when the SM unit requests data from a BMC unit, the BMC unit returns the total size of the requested data in the first response message. This enables the requesting SM unit to keep track on how much data it has received in the response messages and how many request messages that are needed to get the rest of the data. Similar, when the SM unit need to write (*i.e.* send) data to a BMC unit, it will in the first request message include the total filesize. This to help the receiving BMC-unit allocate enough memory for the complete data file. Note that when sending data from the SM unit to the BMC units, the request messages will contain the data while the response messages will be used to acknowledge the last received data part. On the contrary, when data is to be sent from a BMC unit to the SM unit, the data will be transported in the response messages, while the request messages will be used to request them.

VI. Discussion

In this section of the paper we will show some of the limitations to the proposed solution and where future efforts can be made to evolve the presented ideas.

A. Transport performance

It has become obvious during the analysis that the requirements are general with respect to how commands and data are conveyed. *E.g.* there is no particular relation which bus is used for external control. But, the IPMB bus is a well-suited vehicle for the transport of the remote control.

The major drawback with the IPMB bus, and all serial busses, is the speed limit (*i.e.* the limits of the I^2 CBus) of 100 kbit/s and together with a maximum packet size requirement of 25 data bytes (due to maximum overall message duration on the IPMB Bus of 20 ms) hampers the transport of larger tests sets significantly. The bit rate limitation of the bus mostly effects the operation in the

the embedded diagnostic test and the remote diagnostics,the two last test scenarios presented above, since these two require larger amount of data to be transported. However, it is also those two scenarios that might allow a longer test time so this trade-off might be acceptable.

There are faster modes available in the I^2C Bus standard (400 kbit/s in Fast-mode and up to 3,4 Mbit/s in High-speed mode.) However, these faster transfer rates are not currently supported by the IPMI and IPMB specifications, which is one of the requirements to follow in this project.

We will use an example where we are transporting a BVF-file containing a full board interconnection test from the SM-unit to a local BMC unit to illustrate the performance of the IPMB with our extended protocol. The size of the BVF-file is 77 kB and in Table 1 the total transport time of the file is calculated for a few different systems setups regarding the maximum allowed packet size and the chosen I^2C-Bus speed. It is only the first row in Table 1 that follows the requirements set by the IPMI/IPMB specification, however we see that by allowing larger packet sizes on the IPMB and the faster available I^2C-Bus speeds we can improve the total transport time of the IPMB considerably. The price for the increased performance is however reduced availability of the IPMB due to the larger packages on the bus and that we do not follow the requirements set by the IPMI/IPMB specification.

B. Embedded Boundary-Scan management

The command set presented in this paper is in reality only the core of the needed commands in a complete command set. We are aware that it might need modification and expansion to accommodate all the needs in embedded Boundary-Scan tests in a large multi-drop environment. However, one will come a long way by only using the WRITE, DELETE and RUN commands together with the defined embedded binary vector format.

In a more broad perspective where future efforts need to be made is to standardize a common and open command-driven interface which ATE and diagnostic tool vendors could use to develop embedded Boundary-Scan tests more efficiently. Such an interface should have some of the characteristics of the API presented in this paper, especially the modular design and structured command and embedded vector format, to be successful. Today the tools are usually based on low-level, HW dependent, interfaces to access embedded Boundary-Scan paths in systems. One such common approach is a PCI-interface card with multiple TAP ports (a.k.a PODs) which directly link to the Boundary-Scan chains to the units under test. In our, and perhaps in a future solution a standardized interface could be used to access embedded Boundary-Scan tests through one single system test port.

In this study we focused on how system level Boundary-Scan test access could benefit from the IPMI framework. However, in practise any kind of Boundary-Scan data or even any kind of data can be transported using IPMB and our protocol as a vehicle in the backplane. ISP (*i.e.* In System Programming) and remote updating of software comes to mind as possible applications.

VII. DEMONSTRATION

This section contains a description of the basic design of the demonstration board. The board consists of the fundamental parts of the ATCA/IPMI structure and has been implemented with a subset selected of the above presented new functionality. The objective is to display and validate the solution presented in this paper.

The demonstration board is divided into two sides; the left side demonstrates the main shelf management board in an ATCA system and the right side demonstrates one of the application specific boards in the system (Figure 3 and Figure 4). The two sides are using the IPMB Bus for communication as specified in the ATCA specification.

The shelf management side (left) has a microcontroller acting as the Shelf Master (SM) and an external serial port for connection of a PC with management software.

TABLE 1 Transport times of a 77 kB EVF file on IPMB.

Description	Max. packet size	I^2C Speed	Total transport time
Standard IPMB restrictions[a]	32 bytes	12.5 kB/s	11.7 s
Increased maximum packet size	64 bytes	12.5 kB/s	8.44 s
Fast I^2C mode	32 bytes	50 kB/s	2.94 s
I^2C High speed mode	32 bytes	425 kB/s	0.35 s
I^2C High speed mode and increased maximum packet size	64 bytes	425 kB/s	0.25 s

a. This is the only setup that follows the present requirements set by the IPMI/IPMB specification.

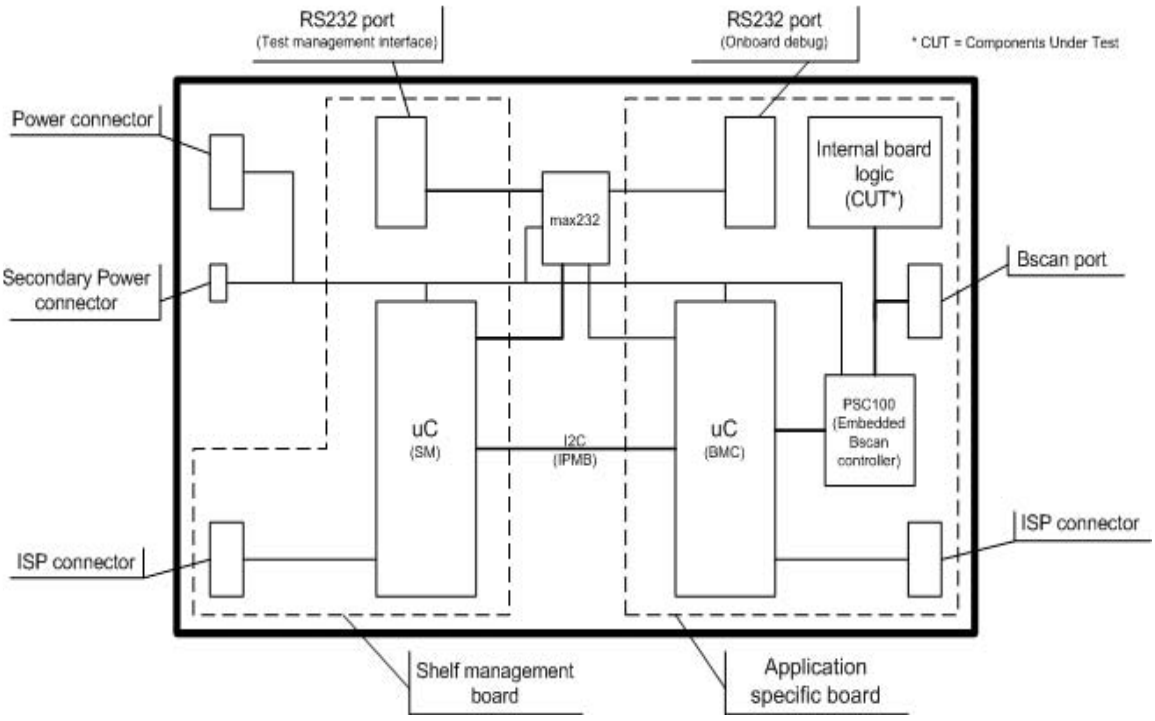

Figure 3: Schematic of the demonstration board.

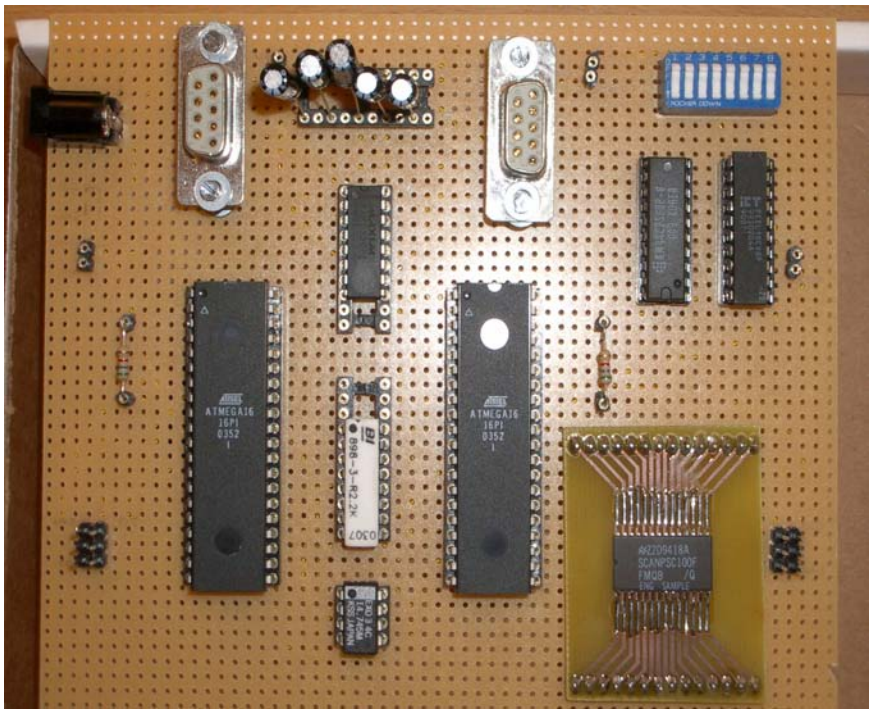

Figure 4: The demonstration board.

The application board side consists of a microcontroller acting as Baseboard Management Controller (BMC), an Embedded Boundary-Scan controller and some internal components acting as targets for the Boundary-Scan tests (Components Under Test, CUT). This provides a complete scan-chain from the Boundary-Scan controller to the CUTs and back. There is also some support parts implemented on both sides, like RS-232 interface components, a dc power connector, debug and in system programming (ISP) ports.

The link between the shelf management board and an external source is not in the scope of this paper, this is specified by the ATCA specification. The RS-232 connection used on the demonstration board was selected for its simplicity and ease to implement.

Below is the list of the main components that was used to design the demonstration board.

- Microcontroller(s): Atmel ATmega16 (8-bit RISC) [17],
- Embedded Boundary-Scan Controller: National Semiconductor SCANPSC100F [14],
- RS-232 interface component: Maxim max232 [18],
- Crystal oscillator (14.7456 MHz): IQD Compact Crystal Oscillator EXO3,
- Components Under Test.

VIII. 8.Conclusions

A major problem in a multi-board system is the limited wiring capability in the backplane. Additional Boundary-Scan wiring to link the boards is therefore highly costly. However, the problem is to access the Boundary-Scan tested boards with the Boundary-Scan controller at the central board. In this paper we propose an approach suitable for the Advanced Telecom Computing Architecture standard where we make use of the existing I^2 C-bus and the Intelligent Platform Management Bus (IPMB) protocol for application of operational tests. We have defined a protocol with commands and responses as well as a test data format for storing test data on the boards to support the remote execution of Boundary-Scan tests. For validation of the proposed approach we have developed a demonstrator.

IX. Acknowledgement

The authors would like to thank Ken Filliter and Bill Aronson at National Semiconductor for valuable discussions and the kind support of hardware and software for the development of the demonstration board.

References

[1] IEEE Standard 1149.1-2001, IEEE Standard Test Access Port and Boundary-Scan Architecture, IEEE Computer Society.

[2] K. P. Parker, "The Boundary-Scan Handbook", Kluwer Academic Publishers, 2001.

[3] Data Sheet, Texas Instruments SN74LVT8996, http://focus.ti.com/lit/ds/symlink/sn74lvt8996.pdf

[4] Application Note, National Semiconductor Application Note 1259 SCANSTA112 Designers Reference, http://www.national.com/an/AN/AN-1259.pdf

[5] PICMG 3.0: AdvanceTCA Core Short Form Specification, www.picmg.org/pdf/ PICMG_3_0_Shortform.pdf

[6] Philips Semiconductor I^2C-bus, http://www.semiconductors.philips.com/acrobat_download/literature/9398/39340011.pdf

[7] Intelligent Platform Management Interface, ftp://download.intel.com/design/servers/ipmi/IPMIv2_0rev1_0.pdf

[8] L. Whetsel, "A Proposed Method of Accessing 1149.1 in a Backplane Environment", *Proceedings IEEE International Test Conference (ITC)*, Baltimore, Maryland, USA, September 1992, pages 206-216.

[9] R. N. Joshi, K. L. Williams, and L. Whetsel, “Evolution of IEEE 1149.1 Addressable Shadow Protocol Devices”, Proceedings of IEEE International Test Conference (ITC), Charlotte, NC, USA, October 2003, pages 981-987.

[10] D. Bhavsar, “An Architecture for Extending the IEEE Standard 1149.1 Test Access Port to System Backplanes", *Proceedings of IEEE International Test Conference (ITC)*, Nashville, TN, USA, October 1991, pages 768-776.

[11] W. Ke, D. Le and N. Jarwala "A secure data transmission scheme for 1149.1 backplane test bus", *Proceedings of IEEE International Test Conference (ITC)*, Washington, DC, USA, October 1995, pages 789-796.

[12] IEEE Standard 1149.5-1995, IEEE Standard Module Test and Maintenance (MTM) Bus Protocol, IEEE Computer Society.

[13] IEEE Standard 1532-2002, IEEE Standard In-System Configuration of Programmable Devices", IEEE Computer Society.

[14] Data Sheet, National Semiconductor SCANPSC100F, http://www.national.com/ds/SC/SCANPSC100F.pdf

[15] Data Sheet, Alliance Semiconductor AS91L1001, http://alsc.com/AS91L1001%20Data%20Sheet.pdf

[16] Serial Vector Format specification, http://www.asset-intertech.com/support/svf.pdf

[17] Data Sheet, Atmel ATmega16 (8-bit RISC), http://www.atmel.com/dyn/resources/prod_documents/doc2466.pdf

[18] Data Sheet, Maxim max232, http://pdfserv.maxim-ic.com/en/ds/MAX220-MAX249.pdf