Microelectronics Journal 42 (2011) 218–232

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

An embedded 1149.4 extension to support mixed-signal debugging ☆

Manuel C. Felgueiras [a,*], Gustavo R. Alves [a], J.M. Martins Ferreira [b]

[a] ISEP/DEE, Rua Dr. António B. de Almeida, 4200-072 Porto, Portugal
[b] FEUP/DEEC, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

ARTICLE INFO

Article history:
Received 30 July 2009
Received in revised form
30 July 2010
Accepted 9 August 2010
Available online 15 September 2010

Keywords:
Built-in test
Diagnostics
IEEE 1149.4
Mixed-signals
Verification

ABSTRACT

Debugging electronic circuits is traditionally done with bench equipment directly connected to the circuit under debug. In the digital domain, the difficulties associated with the direct physical access to circuit nodes led to the inclusion of resources providing support to that activity, first at the printed circuit level, and then at the integrated circuit level. The experience acquired with those solutions led to the emergence of dedicated infrastructures for debugging cores at the system-on-chip level. However, all these developments had a small impact in the analog and mixed-signal domain, where debugging still depends, to a large extent, on direct physical access to circuit nodes. As a consequence, when analog and mixed-signal circuits are integrated as cores inside a system-on-chip, the difficulties associated with debugging increase, which cause the time-to-market and the prototype verification costs to also increase.

The present work considers the IEEE1149.4 infrastructure as a means to support the debugging of mixed-signal circuits, namely to access the circuit nodes and also an embedded debug mechanism named mixed-signal condition detector, necessary for watch-/breakpoints and real-time analysis operations. One of the main advantages associated with the proposed solution is the seamless migration to the system-on-chip level, as the access is done through electronic means, thus easing debugging operations at different hierarchical levels.

1. Introduction

Prototype debugging is one of the development phases of a circuit where design errors and, eventually, structural defects are typically detected, diagnosed and corrected. This activity is traditionally done with the assistance of benchtop equipment directly connected to the circuit under debug. However, the applicability of this traditional equipment has been progressively impaired by trends such as the ever-increasing circuit complexity and miniaturization, and the use of surface mount devices (SMD). These trends had limited effect with respect to analog and mixed-signal (AMS) circuits due to their reduced complexity and number of pins when compared to their digital counterparts. This meant that, at the digital side, the increasing difficulties in doing the structural test of digital printed circuit boards (PCB) led to the proposal, development and market acceptance of mechanisms able to surpass the restrictions imposed by physical access. The strategy followed consisted off embedding on the circuit the mechanisms facilitating that structural test, thus initially leading to proprietary solutions inspired in *level sensitive scan design* (LSSD) [1], and later leading to standard solutions such as the IEEE Std. 1149.1 [2], also known as *Joint Test Action Group* (JTAG) or *Boundary Scan Test* (BST). The BST infrastructure facilitated both structural and the internal test and so the market quickly adopted it, with a large part of its success being supported by its potential use as an electronic access port for debug operations [3,4,5]. In fact, several integrated circuits (IC) made available to the market were specifically designed to increase the controllability/observability levels of digital/analog circuit nodes/buses. This was the case of components belonging to the *System Controllability/Observability Partitioning Environment* (SCOPE™) family of Texas Instruments [6] and of the SCAN™ family of National Semiconductor [7,8], as well as other components proposed by the academic community [9]. Some of these components allowed to partition a circuit into sub-circuits where debugging could be done with test equipment connected through a standard electronic access port, i.e. the test access port (TAP) [5]. This strategy was however clearly insufficient for debug operations in real-time due to the serial nature of the BST architecture (i.e. information is serially shifted through the 4-pin TAP). In this context, components that reuse the

☆ The present paper is a condensed compilation of previous papers published by the authors with new material, namely the methodology for verifying the value of a resistor present on an extended interconnection, using the mixed-signal condition detector. This detector reduces the external test and measurement equipment needed for verifying extended interconnections, as required when only using the mandatory IEEE1149.4 infrastructure, in a similar to how simple interconnections are verified in digital printed circuit boards with IEEE1149.1-compatible components.

* Corresponding author. Tel.: +351 228340532.
E-mail addresses: mcf@isep.ipp.pt (M.C. Felgueiras), gca@isep.ipp.pt (G.R. Alves), jmf@fe.up.pt (J.M. Martins Ferreira).

0026-2692/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2010.08.007

IEEE1149.1 infrastructure for supporting advanced debug operations started to emerge, e.g. the digital bus monitor [10] that implements a logic/signature analyzer at board-level. This component included a digital condition detector that allowed filtering the data captured at board-level, during a certain functional operation period, and subsequently stored in an internal memory. The stored data could then be shifted out, through the TAP, for later analysis with external equipment. The following trend consisted in embedding, inside the IC or system-on-chip (SOC), blocks specifically designed for supporting debug operations, reusing once again the TAP as an interface mechanism. This trend was particularly visible in microprocessors, with several examples provided by industry, e.g. [11], which converged to a standard solution known as NEXUS^{TM} [12]. In this context, the problems associated with the migration of entire digital ICs into SOCs, particularly at the debug phase, benefited from earlier, proven solutions based on electronic access mechanisms.

In the AMS area, the dominant situation is practically the opposite, as this type of circuit is usually designed with restrictive specs, tight tolerance margins, and presents a lower complexity level when compared to their digital counterparts. This means that, when mixed-signal ICs are present in a PCB that also contains digital ICs, the AMS part is a relatively small portion of the entire circuit complexity. The debug phase of this type of circuits benefited from the market appearance of the mixed-signal oscilloscope (MSO) [13,14], which combined the visualization of several tens of pure digital channels with that of 2–4 analog channels, while accepting trigger conditions in the digital, analog, or mixed-signal domains. The intrinsic characteristics of AMS circuits and the lack of a standard infrastructure for accessing its nodes supported the long lasting solution of using physical access for debug purposes. Nevertheless, this type of access was to be progressively replaced by *ad-hoc* and structured electronic access mechanisms [15,16,17]. The publication of the IEEE1149.4 std. [18] raised high expectations in the test/debug community, as this infrastructure was formally presented as the natural extension of the IEEE1149.1 std. to the AMS area, aiming at facilitating structural, parametric and internal tests. However, its adoption has proved slow, with most cited limitations being related to the degradation of the circuit performance [19] and the relative high overhead in ICs of reduced complexity, typically the case in the AMS area. While the first limitation can be minimized with proper design rules [20], the second can only be tackled if the infrastructure proves to be useful for purposes other than the original ones specified in the standard, thus further justifying its inclusion at the IC level. In this sense, it has already been proposed to reuse it for:

- parameter characterization, i.e. V_{OL}, I_{OL}, V_{OH}, I_{OH}, V_{IL}, I_{IL} and I_{IH} [21];
- supporting radio-frequency (RF) measurements [22];
- supporting the test of ADCs and DACs [23];
- remote test and debug of PCBs with AMS components [24];
- monitoring analog signals in automotive environments [25];

Additionally, and following a trend observed in the digital domain, some manufacturers made available to the market IEEE1149.4-compatible ICs targeting the increment of controllability/observability levels of analog nodes in PCBs [26,27].

In this line of evolution, the natural, subsequent step should now be the development of IEEE1149.4-compatible mechanisms offering the possibility to execute, inside the PCB, debug operations similar to those supported by an MSO. Considering this equipment as a combination of a logic analyzer with an oscilloscope, this step would be equivalent to the introduction of the digital bus monitor [10], now for the AMS arena. The experience and limitations arising from the use of such a debug mechanism would allow guiding the subsequent development of specific infrastructures for supporting debug operations in mixed-signal circuits. Any standardization effort built on top of such infrastructures would also benefit the later migration to the SOC level, of AMS circuits. In this sort of AMS components, which assume every-day an increasing importance, the analog part accounts for approximately 2% of the total number of transistors, 20% of the total area, and 40% of the total design effort [28], a direct consequence of the lack of a common platform to support the debug phase. Although the analog/mixed-signal portion is smaller than the digital part, its contribution to the development costs is higher and will continue to grow unless a new debugging paradigm is found. A direct consequence of this fact is the extended time spent on the prototype validation phase, which negatively influences the component's time-to-market (TTM), a key factor for the final product success. The current need for structured mechanisms supporting debug operations in AMS circuits is so crucial that, if not properly addressed in the near future, it may impair the normal development curve of several consumer electronic manufacturers [29]. Another emergent and promising area requiring such mechanisms refers to the use of reconfigurable analog circuits, i.e. field-programmable analog arrays (FPAA) [30,31,32], which call for effective functional verification methodologies. Notice that in the case of field-programmable gate arrays (FPGA), the IEEE1149.1 infrastructure is presently reused for reconfiguration purposes [33] as well as for accessing internal user-defined registers or even the contents of all register elements [34]. In summary, the IEEE1149.1 infrastructure proved particularly useful as a structured mechanism for debug operations in digital circuits, and did also provide the basis for other infrastructures specifically developed for debug purposes. Considering the IEEE1149.4 infrastructure to be the formal extension of its predecessor for mixed-signal components, its reuse should be also analyzed for supporting debug operations in AMS circuits. This is the aim of the present work, which proposes reusing this infrastructure for (1) accessing, in an analog fashion, analog/digital nodes (either corresponding to internal nodes or associated with component pins), and (2) interfacing with a mixed-signal condition detector (MCD), i.e. an internal block specifically designed to support debug operations such as breakpoints/watchpoints or real-time analysis.

The rest of this paper is organized as follows: Section 2 presents a debug model for MS circuits; Section 3 explains how the IEEE1149.4 infrastructure is re-used for accessing the circuit nodes under debug; Section 4 introduces and describes the MCD; Section 5 deals with the validation of the associated debug methodology; Section 6 discusses the impact and cost of the MCD; and, finally, Section 7 concludes the paper.

2. The debug model

Circuit debugging is traditionally done with the assistance of benchtop equipment requiring some sort of access type. Some debug tools are specific of microprocessor-based circuits, such as in-circuit emulators (ICE), while others remain generic, such as logic analyzers, oscilloscopes or multimeters. Although each tool can do/support a large number of debug operations, these can be grouped in a small set of debug operation types. According to the simple debug model illustrated in Fig. 1, any debug operation fits into one of four debug operation types, namely:

- Control, observation and verification (COV) of the circuit state
- Breakpoint/watchpoint
- Step-by-step
- Real-time analysis

COV operations are assumed to be the basic debug operations and are used to control/observe/verify the state of a circuit. Breakpoint/watchpoint, step-by-step and real-time analysis are considered

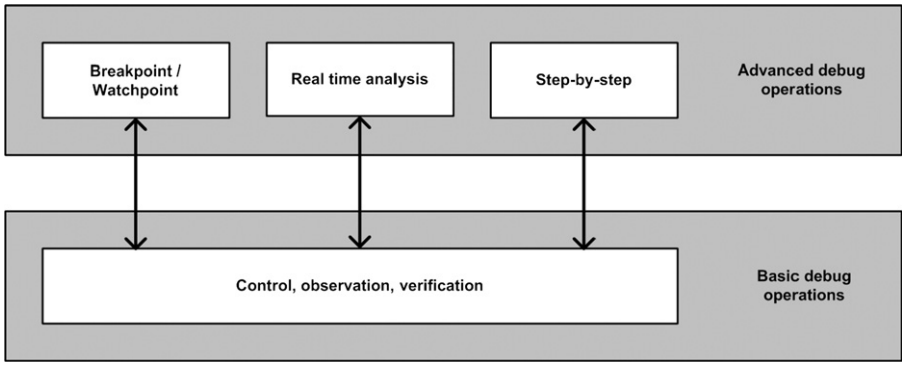

Fig. 1. A simple debug model.

Advanced debug operations and are used to control/observe/verify the circuits function in the time domain.

Providing a simple example of a debug sequence in the digital domain, suppose one intends to verify if a given memory location contains a given value, when the *program counter* reaches a certain address. The sequence of steps is: (1) place a known value in the memory target location, using a control operation; (2) place the circuit in its normal functioning state and then wait till the breakpoint condition is met, i.e. the *program counter* reaches given address, causing the circuit to stop; (3) read the contents of the memory target location, using an operation of the observation type; and (4) verify if the read value matches the expected one, using an operation of the verification type. These debug operations are generic and hence applicable to any sort of digital circuit. For instance, a breakpoint operation can be applied in a sequential circuit by stopping the clock signal, when a certain circuit condition is met, forcing the circuit to retain its present state and enabling the use of COV operations to ascertain about the correctness of that state. A step-by-step operation can be applied to the same circuit to observe and verify each one of its possible states. Real-time analysis operations comprehend state recording and the validation of circuit conditions, in real-time. Typical application examples of this type of debug operation include: store the circuit state *until* breakpoint condition; store the circuit state *after* a breakpoint condition; among others usually supported by logic analyzers, oscilloscopes with memory, and also by MSOs.

In realistic terms, this very basic debug model can be extended to the MS domain. Suppose, for instance, one intends to store the circuit state when an analog value (e.g. a voltage) reaches a given threshold—in basic terms, this would correspond to a breakpoint operation. In the same line of reasoning, a step-by-step operation could be used to verify the cut-off frequency of a digital programmable filter, by applying at the filter input (at each step) an analog signal of increasing frequency and observing (at each step) the filter output. Both debug operation types (breakpoint and real-time analysis) imply the evaluation of circuit conditions and thus require an MCD able to detect them.

The proposed basic debug model and the described debug sequences reveal the need for (1) a mechanism enabling the access to the circuit nodes under debug, and (2) an MCD capable of supporting the referred advanced debug operations. These two aspects will now be addressed in detail in Sections 3 and 4.

3. Extending the IEEE1149.4 infrastructure for COV operations

The access types that distinguish the possible alternatives for implementing COV operations can be divided into: (a) direct physical access; (b) direct electronic access; and (c) non-direct access. Direct physical access comprises the circuit primary inputs/outputs (I/O) and any other circuit access points enabling a direct connection with an automatic test equipment (ATE) or any other test and measurement equipment. This access type presents serious limitations due to the ever-increasing circuit miniaturization levels and the existence of components on both PCB sides – surface-mount technology (SMT) – where some totally prevent the physical access to the device pins, when encapsulated with the ball grid array (BGA) technology. Even when the direct physical access enables observing a given node of the circuit under debug (CUD), it does not imply the possibility to actually control the node value, which depends on the limit for the *backdriving* current. Direct electronic access is done through scan chains or any other dedicated logic/circuit paths. Non-direct access is based on signal propagation through internal circuit blocks, which restricts its practical use to circuits of low/medium complexity.

The IEEE1149.1 and the IEEE1149.4 infrastructures are two examples of the direct electronic access type. While the first is largely used as a mechanism for controlling and/or observing the circuit internal nodes and pins, the second one faces some difficulties for that purpose, according to the study presented in [35]. In fact, the most usual boundary scan cell (BSC) configuration described in the IEEE1149.1 std. (i.e. with two 2:1 multiplexers and two flip-flops) presents a fixed signal flow orientation, allowing at all-times the control of the BSC parallel output, irrespective of the circuit connected to the pin associated with that BSC. In contrast, the switching structure of the analog boundary module (ABM) described in the IEEE1149.4 std. presents a fixed pin orientation. According to the scheme illustrated in Fig. 2a, it is only possible to control, via AT1/AB1, the value of the mission circuit connection (MCC) if the driving source of the pin connection (PC) can be placed in a high-impedance state, i.e. the case, for instance, when the IC containing that driving source is IEEE1149.4-compatible.

Fig. 2b presents a solution to overcome this limitation by adding an extra switch to the ABM switching structure. Notice the IEEE1149.4 std. allows placing additional switches inside that structure.[1] Furthermore, an ABM can also be associated with digital pins to allow controlling/observing the corresponding signals in an analog mode.[2,3] In this line of reasoning, Fig. 3 presents the topology of a proposed generic ABM (G-ABM) that allows controlling/observing the nodes located at both sides of the SD switch.

For a large number of applications, not all elements that form the G-ABM are strictly necessary. However, this a starting point to

[1] (IEEE1149.4—7.3.4): The infrastructure allows adding extra conceptual switches to increase the flexibility of the ABM.

[2] (IEEE1149.4—7.2.1.1.a): The infrastructure allows associating ABMs or DBMs to digital pins.

[3] (IEEE1149.4—3.1.1 NOTE): An ABM may be attached to a digital function pin in order to provide analog measurement capability to the pin.

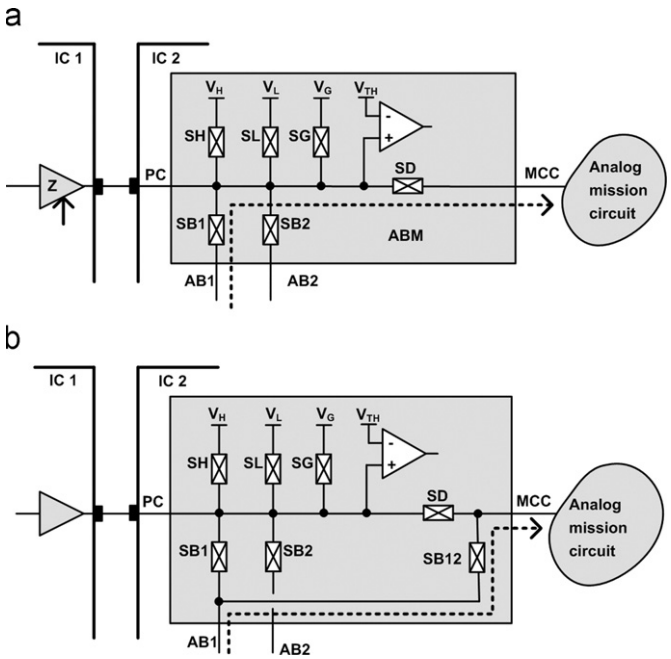

Fig. 2. Modifying the ABM to allow full-control of the mission circuit connection.

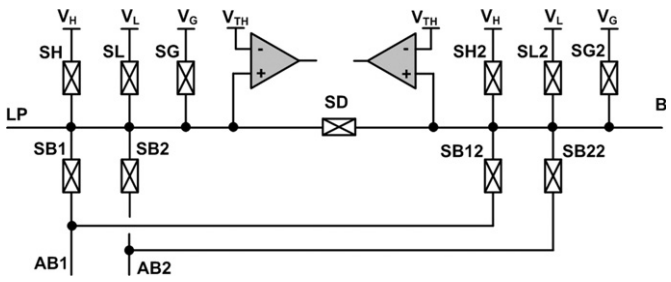

Fig. 3. Topology of the G-ABM.

allow a systematic approach to the needs raised by each and every possible case. The present work proposes additions to the IEEE1149.4 std. allowing (1) the circuit designer to freely choose any ABM resulting from the illustrated G-ABM; (2) the chosen ABMs to be associated with both pins and internal nodes; and, (3) the corresponding control registers to be part of the boundary scan register (BSR). These proposals are twofold. First, they increase the flexibility of the IEEE1149.4 infrastructure, following an approach similar to the IEEE149.1 std., which describes several BSCs that can be selected according to the specific control/observation requirements of each node. Second, they support the claim of conformity with the IEEE1149.4 std. of several circuits using variations of the G-ABM. In fact, conformity will still apply even if partial implementations are used. This is the case, in particular, of several manufacturers that have adopted solutions based on the IEEE1149.4 std., but following internal requirements that led to simpler/reduced modules, in order to minimize the associated overhead [36,37]. In this sense, it is possible to design specific ABM versions targeting the interconnection of internal nodes to the MCD, so as to extend the range of possible breakpoint/watchpoint conditions.

4. The mixed-signal condition detector

From a functional point of view, the MCD compares the value present at the node(s) under debug with one or two limits. As mixed-signal circuits have both digital and analog values, the MCD should support comparison operations in both domains. A detection circuit operating in the digital domain and fully compatible with the IEEE1149.1 infrastructure was already described in [38]. Since this infrastructure uses a serial data transmission protocol, there is a considerable delay between the moment a condition occurs (inside the circuit) and the moment an external detector validates and signals its occurrence. A possible and obvious solution is to move the condition detector into the circuit under debug. There are already proposals of such detection circuits in the analog domain [39,40], which support a reduced number of analog comparison limits and detectable condition types. The proposed solutions also face serious limitations due to their inherent dependence of externally defined analog voltages, corresponding to the comparison limits, which compromise the overall detector precision.

The design of a MCD can follow several directions, according to the requirements identified. In our case, we have considered the following ones:

- Support different comparison operations between observed/expected values.
- Rely on electronic access rather than physical access.
- Guarantee compatibility with the IEEE1149.4 infrastructure.
- Minimize the impact, i.e. the overhead due to the MCD.
- Compare both analog and digital values inside the circuit under debug.

The comparison operations supported by the MCD require an expected value and a mask for operations of the type $=$ and $\neq$, a Limit_A for operations of the type $>$, $\geq$, $<$ and $\leq$, and also a Limit_B for operations of the type $\in$[Limit_A;Limit_B] and $\notin$[Limit_A;Limit_B].

The most efficient solution to overcome the difficulties associated with the physical access is to migrate all the mechanisms needed for doing those operations into the circuit under debug. This has been the most relevant reason to justify the inclusion on the target circuit (i.e. the circuit under debug) of ad-hoc mechanisms, plus standard or proprietary infrastructures. Also in the present case, the best location will be the one allowing access to both digital and analog nodes. Depending on the hierarchical level, that location can either correspond to an IC inside a PCB or a block inside an IC. The trend towards miniaturization, which increasingly promotes the interest for the SOC level, justifies the inclusion of both the MCD, as a built-in dedicated block, and the mechanisms necessary to access any node under debug.

The overhead introduced by the MCD should be minimized through the reuse, whenever possible, of the test resources already available inside the IC. In this sense, it is possible to reuse the IEEE1149.4 test infrastructure to: (1) access this new block; (2) select the nodes under debug; and (3) store, in a digital form, values required for the comparison operations (expected value/mask or Limit_A/Limit_B). The register elements of every BSC – now called digital bus module (DBM) in the IEEE1149.4 – can be used for this purpose, as test operations are not concurrent with debug operations. In contrast, the register elements pertaining to the test bus interface circuit (TBIC) and ABM control structures cannot be reused for this purpose, as the state of the associated switching structures depends on the current contents of those registers. As both the TBIC and ABMs are used to route the signal from the analog node under debug till the MCD analog input, it is not possible to accommodate simultaneously the two concurrent goals.

In order to support detection operations in both domains, the MCD must include two independent condition detectors [41], one for the digital condition and another for the analog condition, as illustrated in Fig. 4.

Each detector has the following inputs: the signal(s) from the node(s) under debug; the expected value/Limit_A; the mask/Limit_B, and a set of configuration lines to define the current operation. The output of each detector is named after its domain, i.e. analog/digital valid condition (AVC/DVC), which is logic high ("1") whenever a condition is evaluated as true. The MCD output, named output valid condition (OVC), can either exhibit the AVC signal, the DVC signal, or a logic combination of both, following a typical functionality present in MSO. The OVC signal can be used for debug purposes inside the CUD (e.g. stop the clock signal on a breakpoint operation) or simply to externally indicate the detection of a valid condition (e.g. on a watchpoint operation). Fig. 5 illustrates a debug scenario where the MCD is used to support a breakpoint operation on an AMS circuit with a microprocessor.

In this sort of circuits (with a microprocessor) it is important to hold the operation at the very moment a condition is met, either in the digital or analog domain. However, holding the circuit operation on the digital/analog domain, upon the occurrence of a condition in the pure analog domain, is not often supported due to the implicit interaction between the two domains.

As the comparison operations of the MCD are done in the pure digital domain, the most direct solution to store all the data needed for those operations is to reuse the memory elements pertaining to the test infrastructure (not simultaneously used for other purposes) and include additional registers accessible through that same infrastructure. These operations take place while the CUD is on its normal operational state, so it is possible to reuse the IEEE1149.4 infrastructure for storing part of the data needed. In fact, the DBM registers can be reused to store the expected value and the mask (or the Limit_A and the Limit_B) – for the condition in the digital domain – on the update (U) and capture/shift (C/S) stages, respectively. In this way, the BSR is reused on its full storage capacity (minimizing overhead), as the memory elements of the TBIC and ABM control registers must hold their previous values. Notice that while the operational mode of a BSC depends mostly on the current test instruction, the operational mode of the TBIC and ABMs also depend on the contents stored in the corresponding control registers that are part of the BSR. In order to store 2 vectors in the U and C/S stages of each DBM, the UpdateDR signal (true at logic "1" when the TAP controller is on the *Update-DR* state) must be deactivated to avoid

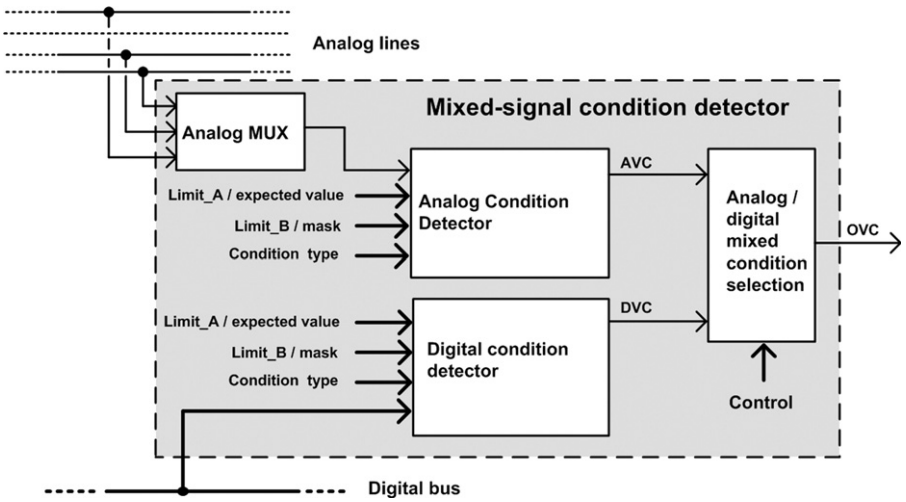

Fig. 4. Conceptual block diagram of the MCD.

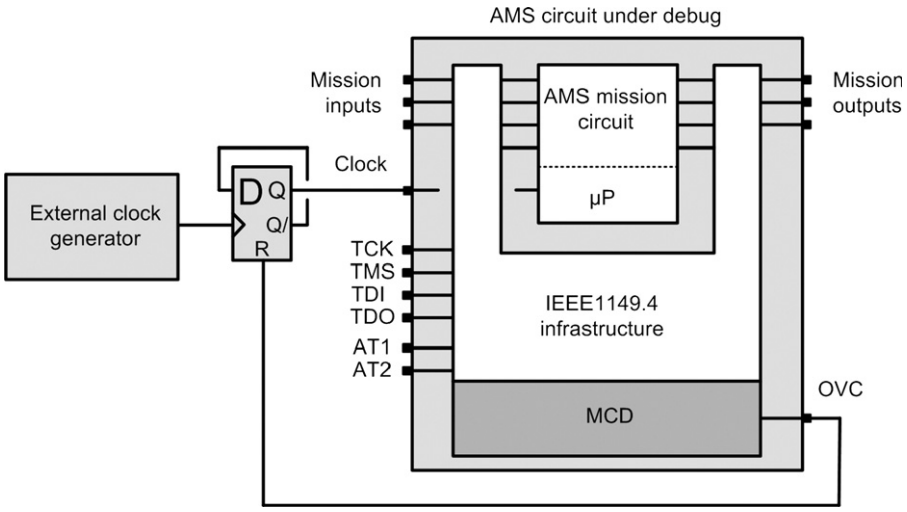

Fig. 5. Using the MCD to support a breakpoint operation.

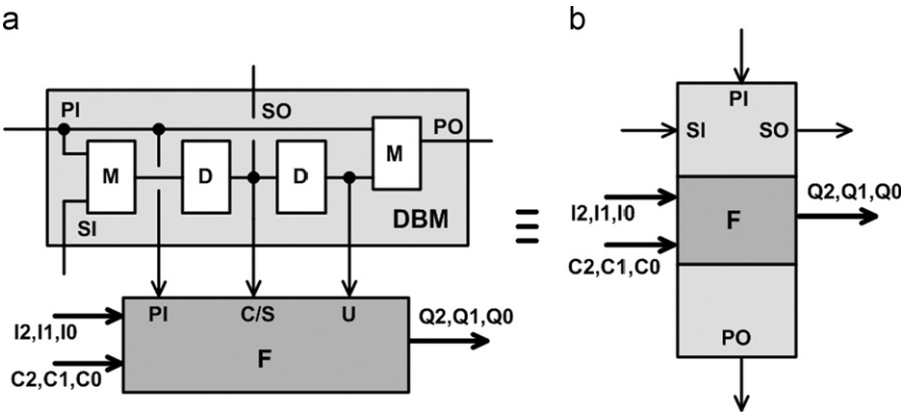

Fig. 6. Structure of a basic (1-bit) condition detector and the corresponding simplified representation.

corrupting the data stored in the Update stage. A DBM can thus contain the 3 signals needed for a comparison operation, as illustrated in Fig. 6.

The structure composed by a single DBM and a single F block forms a basic (1-bit) condition detector, whose output (Q2,Q1,Q0) depends on the:

- actual observed value, present at the parallel input (PI) of the BSC;
- expected value/mask or Limit_A/Limit_B stored in the DBM (C/S, U stages);
- result from the previous basic condition detector (I2,I1,I0); and
- selected comparison operation (C2,C1,C0).

There are five possible comparison results exhibited at (Q2,Q1,Q0): false, true, equal to A, great than A, or less than B; thus 3 lines (2^3) are necessary to codify all possibilities. The F block supports 8 comparison operations, which requires 3 lines to codify them, according to the sequence presented in Table 1.

According to the current operation type (C2,C1,C0), the F block evaluates the result present at (Q2,Q1,Q0), which is a logic function of the inputs (I2,I1,I0) and of the contents present at C/S, U and PI. Table 2 presents, as an example, the truth table for operation "=A", selected when (C2,C1,C0)=(0,0,0). The remaining 7 truth tables are not shown here due to space restrictions.

Fig. 7a illustrates how the several basic condition detectors, each formed by a DBM+an F block, are concatenated to form a digital condition detector register (DCDR), which allows detecting *n*-bit digital conditions. This register acts as a word comparator whose result is shown in the DVC output. The additional FA block is necessary to define, according to the current operation, the input values (I2,I1,I0) for the first basic condition detector. Likewise, the additional FB block is necessary to compute the DVC logic value according to the current operation and the output values of the last basic condition detector. A simplified representation of the DCDR is illustrated in Fig. 7b.

A similar yet simplified register associated with an ADC is used to detect analog conditions. The comparison operation takes place in the pure digital domain, as illustrated in Fig. 8, with the ADC input connected to one line (AB2) of the internal analog test bus—a part of the IEEE1149.4 infrastructure. The analog condition detector register (ACDR) is an optional register, permitted by the IEEE1149.4 std., internally built with DBM-alike blocks without the second multiplexer (i.e. it does not have primary outputs), as it only receives information from the ADC.

The operation type implemented by the analog condition detector register (ACDR) and the DCDR is defined by inputs (C2A,C1A,C0A) and (C2D,C1D,C0D), respectively, fed by an equal

Table 1
Comparison operation types supported by the MCD.

C2	C1	C0	Operation type	Note
0	0	0	=A	Requires a mask
0	0	1	≠A	Requires a mask
0	1	0	> Limit_A	
0	1	1	< Limit_A	
1	0	0	≥ Limit_A	
1	0	1	≤ Limit_A	
1	1	0	Inside the interval [A, B]	Requires a Limit_B
1	1	1	Outside the interval [A, B]	Requires a Limit_B

Table 2
Truth table of the F block for the "=A" operation.

Coded (I2,I1,I0)	C/S	U	PI	Coded (Q2,Q1,Q0)
F	X	X	X	F
T	0	X	X	T
T	1	O	0	T
T	1	O	1	F
T	1	1	0	F
T	1	1	1	T

number of bits belonging to the detection configuration register (DCR). The FC block is responsible for driving OVC (an MCD primary output), from 4 possible sources: (i) AVC, (ii) DVC, (iii) AVC AND DVC and (iv) AVC OR DVC. The source is selected through signals VS1 and VS0, i.e. 2-bits belonging to the DCR. OVC is valid when inputs COMP2 and RTI are true. These two signals act as an enable input. COMP2 is true at logic level "1" when the IEEE1149.4 instruction register (IR) is loaded with the optional instructions *EXTEST2*, *PROBE2* or *INTEST2* (described ahead), while RTI is true at logic level "1" when the TAP controller is on the *Run-Test/Idle* state. Table 3 resumes the operation of the FC block.

OVC can either be an internal signal used for breakpoint operations, when the necessary mechanisms for stopping the circuit operations are present, or a dedicated output pin used to flag an event to the outside world, as already illustrated in Fig. 5.

DCR corresponds to an 8-bit optional register added to the registers structure accessible through the IEEE1149.4 infrastructure. Fig. 9 illustrates its internal contents, where the purpose of each group of bits was already described in the previous paragraph.

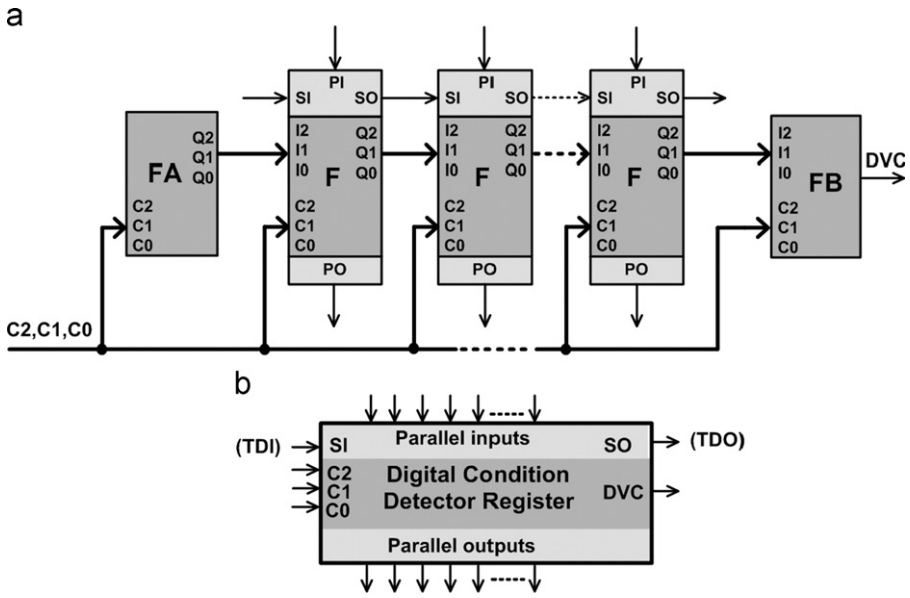

Fig. 7. DCDR structure and simplified representation.

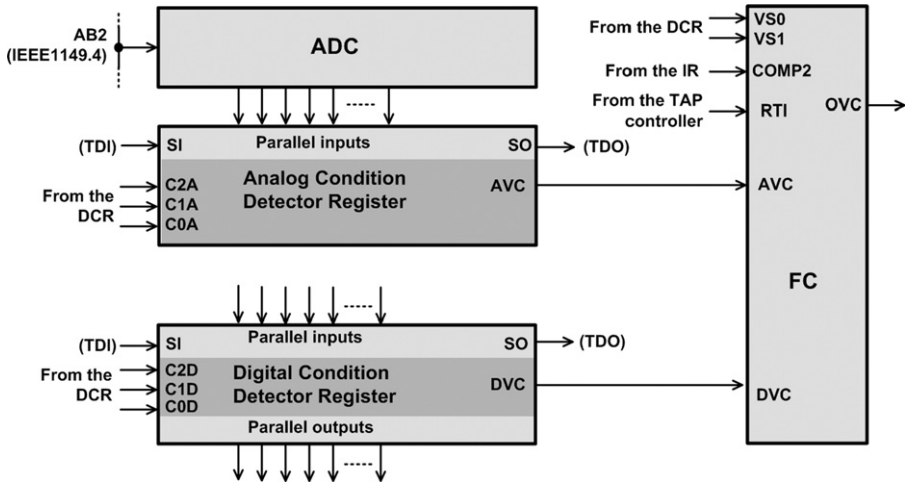

Fig. 8. MCD structure.

Table 3
Truth table for the OVC signal.

RTI	COMP2	VS1	VS0	OVC
0	X	X	X	0
X	0	X	X	0
1	1	0	0	DVC
1	1	0	1	AVC
1	1	1	0	DVC **OR** AVC
1	1	1	1	DVC **AND** AVC

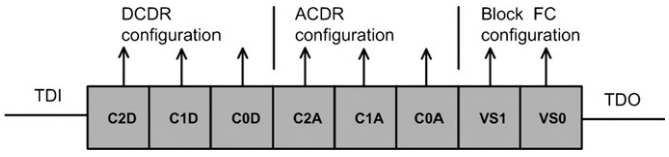

Fig. 9. The DCR description.

The DCDR is formed by the DBMs already present in the circuit, making this register a part of the BSR. The same does not apply to the ACDR. However, it should be possible to shift in the Limit_A vectors to both registers, during the same shift sequence, and then store them at the respective update stages. The same should be also possible for the Limit_B vector. The register structure presented in Fig. 10 enables this possibility by serially connecting the boundary scan and the analog condition detector registers (i.e. BSR+ACDR), when the data multiplexer input 2 is selected. Selecting input 1 enables accessing the DCR for debug configuration purposes, while the two remaining inputs (3 and 0) serve the mandatory boundary scan and bypass registers. Notice that DCDR is not represented as a discrete register as it is part of the mandatory BSR.

In order to operate the functionally provided by the MCD, it is necessary to add a number of new optional instructions to the IEEE1149.4 infrastructure. A first optional instruction, named *SELCON*, is required to place the DCR into the test data input (TDI)—test data output (TDO) path, i.e. to select input 0 of the data mux illustrated in Fig. 10. A second optional instruction,

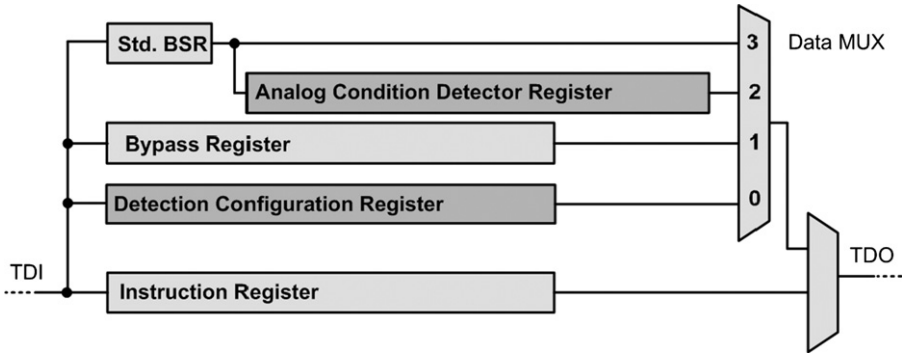

Fig. 10. Registers structure supporting the MCD.

Table 4
Characteristics associated with the proposed optional instructions.

Optional instruction	Selected data mux input	*UpdateDR* signal disabled	Behaviour of the remaining IEEE1149.4 infrastructure
PROBE2	2	Yes	=*PROBE*
INTEST2	2	Yes	=*INTEST*
EXTEST2	2	Yes	=*EXTEST*
S/P2	2	No	=*S/P*
SELCON	0	–	=*BYPASS*

named *SAMPLE/PRELOAD2* (or *S/P2*, in an abbreviated form) selects input 2 of the same data mux, to allow storing the Limit_A vector on the U stages of DCDR and ACDR. A third optional instruction, named *PROBE2*, has multiple purposes: (1) it selects input 2 of the data mux to allow storing the Limit_B vector on the C/S stages of DCDR and ACDR; (2) it disables the *UpdateDR* signal to avoid corrupting the previously stored Limit_A, when the TAP controller passes through the *Update-DR* state; (3) it allows selecting the current analog node under debug feeding the ADC input, i.e. in a similar operating mode provided by the mandatory *PROBE* instruction, it connects the AB2 switch of the ABM associated with the analog node selected for debug purposes and routes the signal through the internal test bus line with the same name (AB2) to the input of the ADC associated with the ACDR. Similar to the *PROBE* mandatory instruction, the *PROBE2* optional instruction allows using the internal analog test bus in a non-intrusive mode during the mission circuit normal operation mode. Another two optional instructions, named *INTEST2* and *EXTEST2*, allow using the MCD when the IEEE1149.4 infrastructure is placed in the internal and external test modes, respectively. When the present instruction is PROBE2, *INTEST2* or *EXTEST2*, the remaining elements of the IEEE1149.4 infrastructure retain their operational conditions, defined for the mandatory instructions PROBE, INTEST and EXTEST, respectively. Table 4 summarizes some characteristics of the proposed optional instructions.

The following section will now describe the sequence of IEEE1149.4 operations necessary for using the MCD and the G-ABM to implement part of the debug model presented in Section 2.

5. Validating the proposed solution through simulation

To validate the proposed debug model it is necessary to have a mission circuit surrounded by an IEEE1149.4 infrastructure comprising the previously defined extensions, i.e. the MCD and the non-canonical ABM topologies that allow (i) controlling the mission circuit inputs in an independent mode, and (ii) accessing internal analog nodes for control/observation purposes. The selected mission circuit should also allow demonstrating the use of both analog and mixed-signal condition detection operations.

Developing a circuit in hardware with such characteristics would be quite time consuming, as the number of IEEE1149.4-compatible devices is reduced and, furthermore, do not support the proposed ABM variants. The alternative of implementing a compatible circuit, using discrete components, is also quite cumbersome and time consuming as already demonstrated through the example described in [42], referring to a simple functional circuit composed of a digital inverting buffer and an operational amplifier with unitary gain, surrounded by the mandatory IEEE1149.4 infrastructure. In order to validate the proposed debug model, a substantially more complex circuit is required, thus favouring the use (in a first approach) of a simulation environment. We decided for OrCAD v10.3, supported by Cadence, which includes a PSpice simulator. The main characteristics of the mission circuit designed for validation purposes are:

- Includes at least two mixed-signal macro blocks, enabling the association of an ABM to an internal analog node.
- Enables detection operations on analog nodes and/or digital buses.

As illustrated by Fig. 11, the selected macro-blocks composing the mission circuit are a 4:1 analog mux and a 12-bits ADC (named ADC1). ADCs and analog multiplexers/switches are quite common in MS circuits, with the association of ADC plus analog mux, in particular, being frequently found in circuits comprising a micro-controller.

The complete target circuit includes the mission circuit plus an IEEE1149.4 infrastructure with the proposed extensions, i.e. the ABM variants described in Section 3, and the MCD described in Section 4. The main characteristics of the target circuit are:

- The digital pins have DBMs associated with.
- The analog input pins have ABMs associated with a topology corresponding to the one illustrated by Fig. 2b (identified as variant ABM-1).

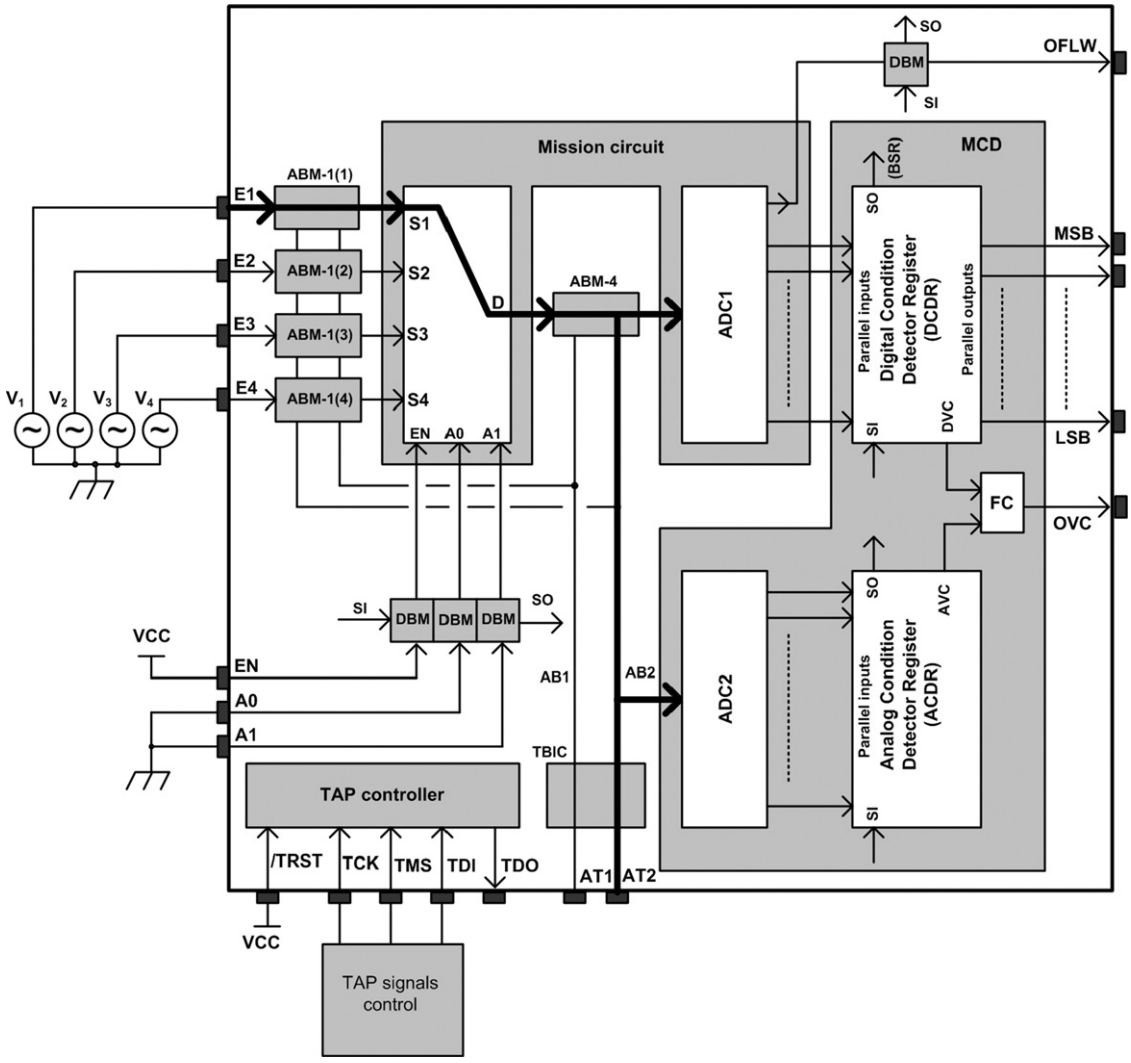

Fig. 11. Mixed-signal target circuit used for verification purposes.

- The internal analog node corresponding to the interconnection between the analog mux output and the functional ADC input contains an ABM whose topology includes the SB1, SD and SB22 conceptual switches of the proposed G-ABM (see Fig. 3). This topology is named ABM-4 and allows full control/observation of the analog node through the internal test bus AB1/2 lines.
- The IEEE1149.4 std. imposes DBMs for the functional ADC outputs. These DBMs are reused in the DCDR.
- The MCD and the associated ADC (named as ADC2) are part of the internal debug and test infrastructure and allow detecting analog conditions (defined by the ACDR).
- ADC1 and ADC2 are equal and present a transfer function where the analog input range [−10 V; +10 V] is converted into a digital word in the [000h; FFFh] interval.

To validate part of the proposed debug model we follow an example where the MCD is used with the "> Limit_A" and "< Limit_A" operations, for detecting an analog and a digital condition, respectively, during the circuit normal operation. The circuit is placed in the following normal operating conditions:

- Four different analog signals (V_1–V_4), simulating four different analog sources, are applied to the circuit functional inputs *E*1–*E*4, respectively (corresponding to the analog mux inputs *S*1–*S*4). The associated ABMs should be in the transparent mode.
- The digital control inputs (EN,A1,A0) of the analog mux should be controlled through the pins holding the same name. The associated DBMs should thus be in the transparent mode. In the provided example, input S1 is driving the analog mux output D, which implies (EN,A1,A0)=(1,0,0).
- ABM-4 should be in the non-intrusive observation mode (i.e. switches SD and SB22 closed, and SB1 open), allowing the signal present at the analog mux output D to be routed to the ADC2 input, through AB2.
- OVC should be at logic level high ("1") whenever the internal analog signal present at the interconnection between the analog mux output and the functional ADC input is higher than +6 V **OR** the digital output pins of the mission circuit (corresponding to the ADC1 data bus) exhibit a vector value lower than 66Bh.

To select this debug operation, we first define the DCR contents, i.e. the values uploaded into (C2D,C1D,C0), (C2A,C1A,C0A) and (VS1,VS0). The first two sub-vectors are defined according to Table 1 and the last one according to Table 3. In the present example, we select the "< Limit_A" operation for the digital part, which corresponds to (C2D,C1D,C0)=(0 1 1); the "> Limit_A" operation for the analog part, which corresponds to (C2A,C1A,C0A)=(0,1,0); and OVC as a logic **OR** between AVC and DCV, which corresponds to

Table 5
ADC1 and ADC2 conversion values.

ADC (analog) input value (V)	ADC (digital) output value	Comments
−10	000h	Lowest value
−2	66Bh	Digital condition value
0	800h	(GND)
+6	CD7h	Analog condition value
+10	FFFh	Highest value

(VS1,VS0)=(1,0). This results in shifting in the (C2D,C1D,C0,C2A, C1A,C0A,VS1,VS0)=(011′010′01)=6Ah vector into DCR, using the *SELCON* optional instruction. Next, we must select the values corresponding to the analog Limit_A and to the digital Limit_A. The first corresponds to the digital conversion of the indicated upper limit voltage (+6 V), i.e. (110011010101)=CD7h, while the second is already a digital word, i.e. (011001101011)=66Bh. These two analog values / digital words are represented in Table 5 for the sake of clarity.

The two digital words are shifted into DCDR and ACDR using the optional *S/P2* instruction. The "< Limit_A" and "> Limit_A" comparison operations do not require a Limit_B nor a mask, so we may shift an all 1's vector to the C/S stages of those two condition detector registers, using the *PROBE2* optional instruction. The detection process starts the moment the TAP controller enters the *Run-Test/Idle* state. The following paragraph lists the pseudo-code with the sequence of steps necessary to configure the IEEE1149.4 infrastructure and the MCD with the previously identified parameters.

```
Instruction Register ← SELCON;
  %Select position 0 in the Data MUX (see Fig.10);
Detection Configuration Register
  (C2D,C1D,C0D,C2A,C1A,C0A,VS1,VS0) ← (6Ah);
  %Shift in the vector that selects the analog and
  %digital conditions types, and selects the analog
OR the
  %digital detections to be outputted at the OVC pin;
Instruction register ← SAMPLE/PRELOAD2;
  %Select position 2 in the Data MUX (see Fig. 10);
SR+Analog Condition Detector Register ← (YYY…66B
CD7h);
  %Shift in the vector that selects the analog node
  %under analysis (YYY) and the Limit_A for the
Digital (BSR) and
  %Analog Condition Detector Registers; the digital
  %value (that corresponds to -2 V on the analog
value)
  %is shifted into the Digital Part; the analog value
  %that corresponds to +6 V is shifted into the
Analog part;
Instruction Register ← PROBE2;
  %Select position 2 in the Data MUX (see Fig. 10);
BSR+Analog Condition Detector Register ← (YYY…FFF
FFFh);
  %Signal UpdateDR disabled in the Analog and
Digital
  %Detection Registers;
  %Shift in the vector that selects the analog node
under
  %analysis (YYY) and the Limit_B for the Digital and
  %Analog Condition Detector Registers. As the
  %selected operations (<Limit_A, >Limit_A) do
not depend of
  %limit_B/mask, an all 1's vector is shifted into
the Digital and
  %Analog Parts.
TAP controller ← Run-Test/Idle;
```

Simulating the described circuit in the OrCAD simulation environment requires the definition of the input stimuli, namely for the TAP input signals (TCK, TMS and TDI). To speed up the simulation process, we developed an in-house application, named BSOrcad, which automatically generates the input stimuli for the 3 digital input signals, from a test program written with commands/data similar to those specified by the serial vector format. The output of the BSOrcad application is a file "<filename>.stl" directly interpreted by the PSpice simulator. Fig. 12 illustrates the simulation results for the example described earlier.

The names appearing at the upper left corner of Fig. 12, correspond to the following signals (by order of appearance, i.e. top-down):

- TCK (test clock), TMS (test mode select), TDI and TDO
- TAP controller state (coded according to the suggestions of the IEEE1149.1 std.)
- IR contents
- DCR contents
- AVC signal
- DVC signal
- OVC signal
- Analog signal present at AB2 (IEEE1149.4)

A careful analysis of Fig. 12 reveals the TAP controller to be in the *Shift-IR* (Ah) and *Shift-DR* (2h) states for longer periods, as it is in these states that vectors are serially shifted into/out of the IEEE1149.4 infrastructure. The last TAP controller state (from ≈3.4 ms onwards) corresponds to *Run-Test/Idle* (Ch), where the MCD will be active (see Table 3), given the instruction loaded into the IR (*PROBE2*, i.e. 06h). The IR is loaded with *BYPASS* (FFh), on power-up, and then loaded with *SELCON* (08h), *SAMPLE/PRELOAD2* (05h), and, finally, with *PROBE2* (06h). The DCR is loaded with 0h, on power-up, and then with 6Ah, according to the explanation provided in the previous paragraphs. After shifting in the vector corresponding to the *PROBE2* instruction (06h), more precisely on the *Update-IR* state (at the TCK falling edge), the AB2 line starts to follow the voltage present at the ABM-4 input. Notice that, in the example provided, AB2 monitors the analog value present at the analog functional input of ADC1 (part of the mission circuit) while DCDR is loaded with the condition to be detected on the ADC1 digital output bus. This combination allows us to verify the detection limits (<Limit_A , >Limit_A) of both analog and digital parts, in relation to the analog voltage present at AB2. The ACDR output (i.e. AVC) is at logic "1" whenever the voltage at AB2 is higher than +6 V, while the DCDR output (i.e. DVC) is at logic "1" whenever the contents of the ADC1 digital output bus is lower than 66Bh, i.e. whenever the voltage at AB2 is lower than −2 V. OVC is at logic "1" when the following conditions are true at the same time: (i) the result of AVC **OR** DVC is true; (ii) the current instruction is *PROBE2*; and (iii) the TAP controller is on *Run-Test/Idle*. The example given illustrates the MCD being used in a debug operation while the mission circuit is on its normal operation mode. This was accomplished with the *PROBE2* optional instruction.

The *INTEST2* optional instruction allows supporting debug operations while the target circuit is on the internal test operation mode, as described in [43]. In this mode, the state of the digital inputs of the mission circuit is defined by the contents present on the U stage of the associated DBMs (on control mode), while the ABMs may either be on the transparent or control mode, although

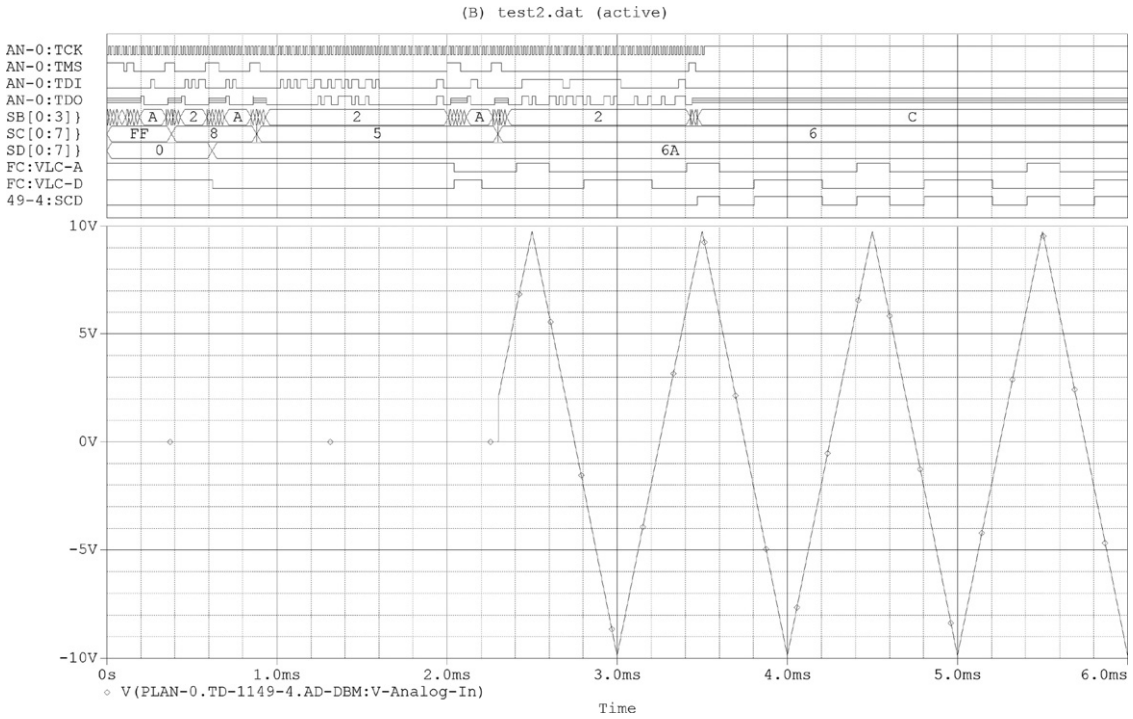

Fig. 12. Detecting an AMS condition during the target circuit normal operation.

the latter requires them to support the resources described on Section 3 (see also footnotes 2 and 3).

The following example illustrates how the *EXTEST2* instruction can be used to verify if a resistor value is within its tolerance window,[4] i.e. it demonstrates the use of the MCD during the external test mode. In the example provided, a resistor connected between the E1 pin and GND is verified. In a broad sense, it could be the case of verifying an extended interconnection between two IEEE1149.4-compatible ICs, or verifying the value of a resistor necessary for configuration purposes, e.g. a configurable IP analog filter used in an SOC.

The verification sequence includes the following steps:

- A known current is injected into R_x, via AT1/AB1.
- The voltage drop at R_x is applied to the MCD analog input, via AB2.
- MCD detects if the voltage present at AB2 is within the voltage limits corresponding to the resistor tolerance range.

Fig. 13 illustrates (in bold) the current path from the AT1 pin to the E1 pin (routed via AB1) and the voltage path from the E1 pin to the MCD analog input (routed via AB2).

In this example, the IEEE1149.4 infrastructure is configured in such a way that: ABM-1(1) has SB1 and SB2 closed and the remaining switches open; ABM-1(2), ABM-1(3) and ABM-1(4) have all switches open; ABM-4 has SD closed and the remaining switches open; the TBIC has switches S5 and S6 closed and the remaining ones open. The resistor value is $47 \pm 5\%$ kΩ and the injected current is 100 μA, which implies a voltage drop in between 4.98 and 4.50 V (corresponding to the BFBh and B9Ah codes, respectively). We select the "$\in$[A, B]" operation for the analog part, which corresponds to (C2A,C1A,C0A)=(1,1,0), while the operation for the digital part is irrelevant, as it will not be used in this verification process, so we select a default value, i.e. (C2D,C1D,C0D)=(0,0,0). The MCD result does not depend on the analog part, so we select (VS1,VS0)=(0,1). The DCR contents will thus be (C2D,C1D,C0D,C2A,C1A,C0A,VS1,VS0)=(0,0,0,1,1,0,0,1)=19h. The following paragraph lists the pseudo-code with the sequence of steps necessary to configure the IEEE1149.4 infrastructure and the MCD with the previously identified parameters.

```
Instruction Register ← SELCON;
  %Select position 0 in the Data MUX (see Fig. 10);
Detection Configuration
Register(C2D,C1D,C0D,C2A,C1A,C0A,VS1,VS0) ← (19h);
  %Shift in the vector that selects the analog and
digital condition
  %types, and selects only the analog detection to be
outputted at OVC;
Instruction Register ← SAMPLE/PRELOAD2;
  %Select position 2 in the Data MUX (see Fig. 10);
BSR+Analog Condition Detector Register ← (YYY…FFF
B9Ah);
  %Shift in the vector that selects the analog node
  %under analysis (YYY) and the Limit_A for the
Digital (BSR) and
  %Analog Condition Detector Registers;
  %for the digital part an all 1's vector is shifted in;
  %for the analog part is shifted in the vector B9Ah
that
  %corresponds to +4.50 V;
Instruction Register ← PROBE2;
  %Select position 2 in the Data MUX (see Fig. 10);
BSR+Analog Condition Detector Register ← (YYY…FFF
BFBh);
  %Signal UpdateDR disabled in the Analog and Digital
  %Detection Registers;
  %Shift in the vector that selects the analog node
  %under analysis (YYY) and the Limit_B for the
Digital (BSR) and
  %Analog Condition Detector Registers;
  %for the digital part an all 1's vector is shifted in;
```

[4] Not previously published—see introductory note.

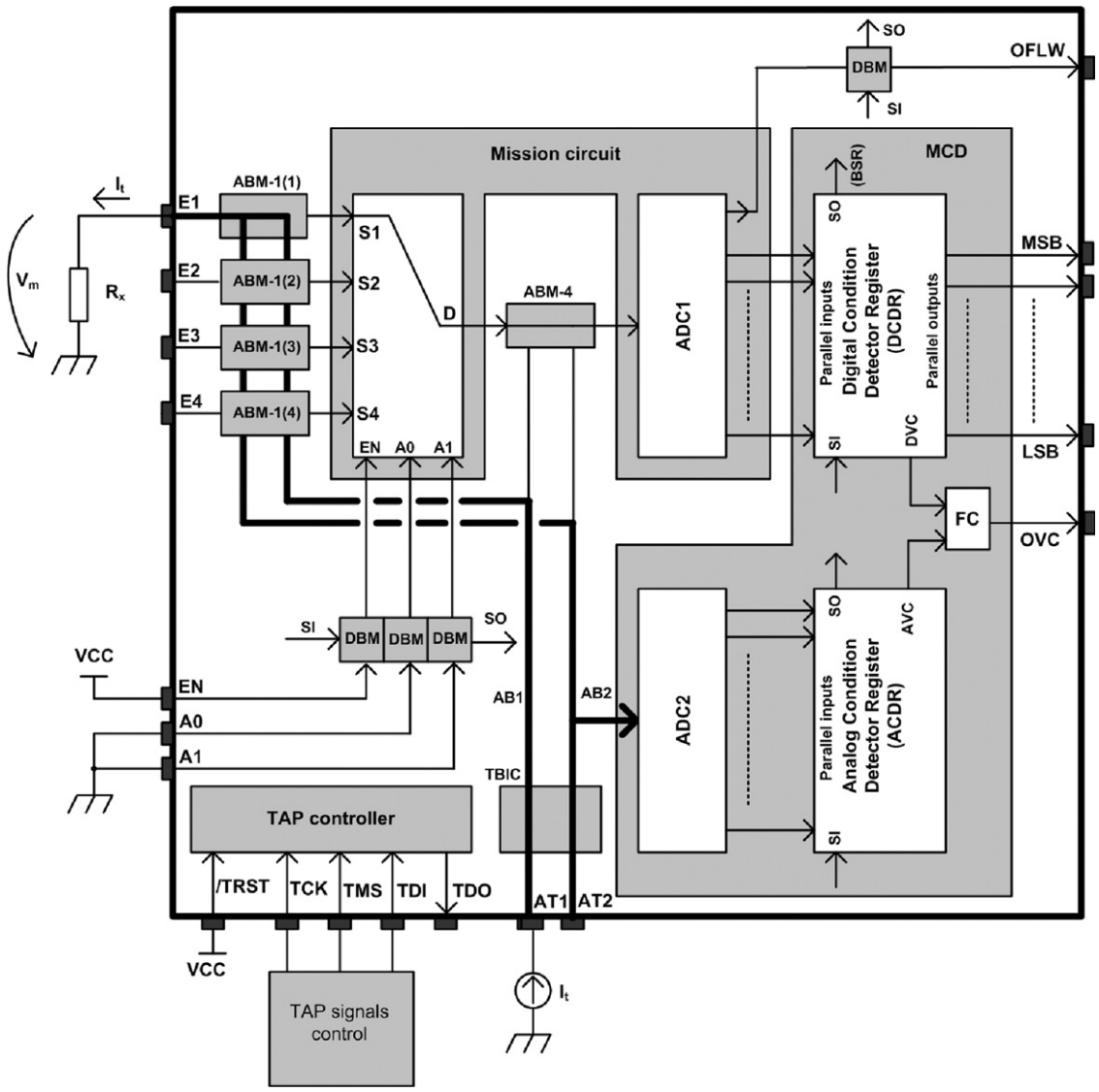

Fig. 13. Verifying a resistor value during an external circuit test.

```
%for the analog part is shifted in vector BFBh that
%corresponds to +4.98 V;
TAP controller←Run-Test/Idle;
```

The simulation results are depicted in Fig. 14, where the names appearing at the upper left corner correspond to the following signals (by order of appearance, i.e. top-down):

- TCK, TMS, TDI and TDO
- TAP controller state
- IR contents
- DCR contents
- Contents of the ACDR C/S stage
- Contents of the ACDR U stage
- Digital output bus of the ADC2 pertaining to the MCD
- OVC signal
- Analog signal present at the AB2 line, part of the internal analog test bus

Fig. 14 reveals how the TAP controller exhibits a sequence of states quite similar to the one occurring on the previous example. The IR is again loaded, on power-up, with the *BYPASS* instruction code (FFh), and then, by chronological order, with the *SELCON* (08h), *S/P2* (05h) and *EXTEST2* (04h) instruction codes. The DCR, presenting by default the 0h value, is loaded with a 19h pattern, as intended. On *Update-IR*, after *EXTEST2* (04h) has been shifted into the IR, the AB2 line starts to exhibit the analog voltage present at the ABM-4 input. The C/S and U stages of the ACDR are loaded with BFBh and B9Ah, respectively. The digital output bus of ADC2 (belonging to the MCD) presents the BC6h pattern when the current instruction becomes *EXTEST2* (04h), where this digital pattern corresponds an analog value of 4.72 V, resulting from the voltage drop at the resistor under verification. A quick inspection on the product given by the injected current times the resistor range (Ohm' law with 100×10^{-6} A [44650, 49350]Ω=[4.47, 4.94] V) indicates a resistor value within the tolerance range. OVC is at logic "1" when the following conditions are true at the same time: (i) the resistor voltage drop is within the specified analog limits; (ii) the current instruction is *PROBE2*; and (iii) the TAP controller is on *Run-Test/Idle*.

The examples described the advantages associated with the use of the MCD for: (i) detecting a mixed condition during the mission circuit normal operation mode; (ii) detecting an analog condition during the internal test operation mode; and, (iii) detecting an analog condition during the external test mode. The following section will now analyze the limitations and the

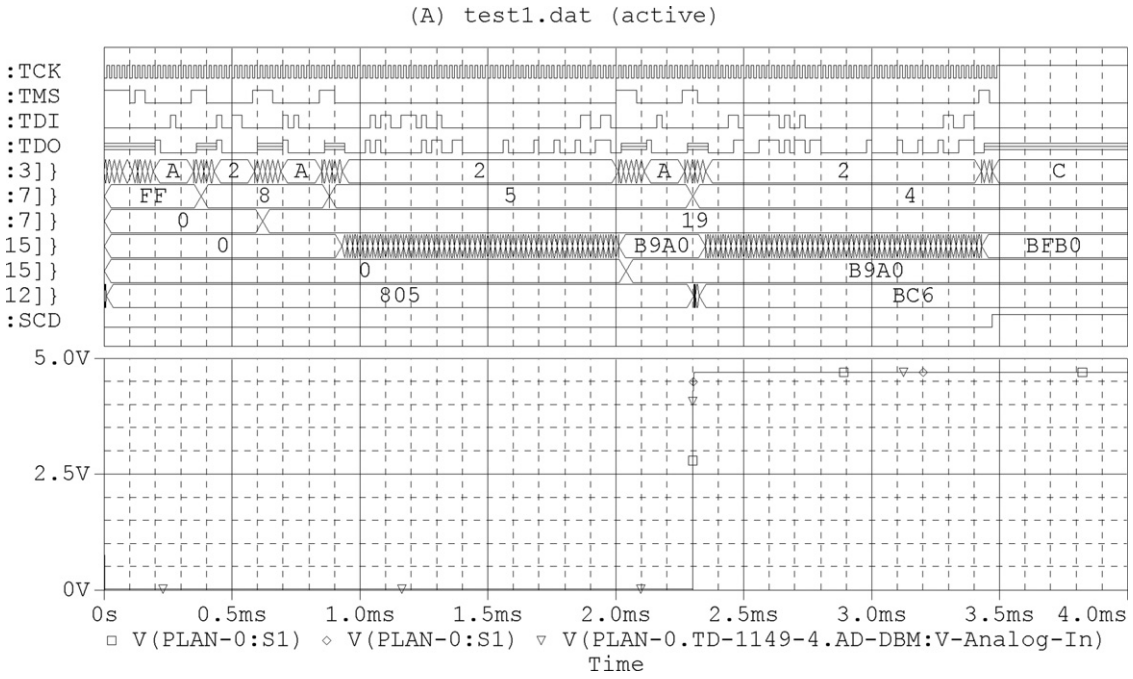

Fig. 14. Verifying a resistor value (within its tolerance range) during an external test sequence.

cost (*overhead*) associated with the proposed built-in debug mechanism.

6. Disadvantages/overhead

The major limitations associated with the MCD are due to the introduced overhead and its functional restrictions. The first results from the inclusion of this block and the resources required by it (e.g. additional ABMs or ABM variants), which may be quantified in terms of an extra pin and the extra silicon area. The last will now be addressed in detail.

Since the silicon area required by each element is technology-dependent, the analysis will consider the *circuit complexity*. In respect to the digital part, the following assumptions are made:

- All circuits can be decomposed into elementary two-input logic gates (G2).
- All elementary two-input logic gates (G2) are equally complex.

Table 6 resumes the complexity of some digital blocks constituting the mandatory IEEE1149.4 infrastructure, based on these two assumptions.

Besides a digital part, the IEEE1149.4 infrastructure also includes analog switches and comparators, not mentioning the ADC2 when the MCD is present. The silicon area required by these three component types is also technology-dependent while the analog nature impairs them to be decomposed, so we decided to keep them apart. To allow a better understanding of the overhead associated with our solution, we will first characterize the complexity of an IEEE1149.1 infrastructure, follow to the characterization of the expansion to an IEEE1149.4 infrastructure, and then consider the inclusion of the proposed MCD.

A test infrastructure has two parts: one fixed and another that depends on the number of DBMs and ABMs. The fixed part includes the following non-replicated blocks: the TAP controller, the IR (with 8 bits, in our case), the instruction decoder, the *Bypass* register, etc. The

Table 6
The complexity of some IEEE1149.4 internal blocks expressed in terms of G2.

IEEE1149.4 blocks	Complexity $N_{(G2)}$
8-bits instruction register	296
Data mux	4
Bypass register	17
Instruction decoder	62
DBM	41
ABM (digital part only)	165

equivalent complexity of this IEEE1149.1 infrastructure is given by[5]

$$N_{(G2)} = 556 + 41N_{DBM}$$

where N_{DBM} represents the number of DBMs included in the infrastructure. For instance, the equivalent complexity of an IEEE1149.1 infrastructure for an IC with 100 I/O pins is $N_{(G2)}=4656$ elementary two-input logic gates. In respect to an IEEE1149.4 infrastructure, the digital part can also be expressed in G2 terms, while the analog part may be expressed in terms of switches and comparators. The equivalent complexity will thus be given by

$$N_{(G2,SWITCHES,COMPARATORS)} = (746 + 165N_{ABM} + 41N_{DBM}, 10 + 6N_{ABM}, 2 + N_{ABM})$$

where N_{ABM} represents the number of ABMs included in the infrastructure. In this sense, the equivalent complexity of an IEEE1149.4 infrastructure with 100 DBMs and 5 ABMs, will be of $N_{(G2,SWITCHES,COMPARATORS)}=(5671,40,7)$. Adding an MCD to the infrastructure (and not considering the ADC2 at the present time) will lead to an equivalent complexity of

$$N_{(G2,SWITCHES,COMPARATORS)} = (1091 + 165N_{ABM} + 41N_{DBM} + 119N_{DREG} + 115N_{AREG}, 10 + 6N_{ABM}, 2 + N_{ABM})$$

where N_{DREG} and N_{AREG} represent the number of bits in the DCDR and in the ACDR, respectively. Considering the existence of the ADC2, the

[5] These values were extracted from the model used in our simulation case.

Table 7
Infrastructure complexity.

		1149.1	1149.4	1149.4+MCD
Circuit characteristics	# of digital I/O	100	100	92
	# of analog I/O	–	5	5
	# bits of the DCDR	–	–	8
	# bits of the ACDR	–	–	8
Infrastructure complexity	# of equivalent G2 gates	4650	5671	7650
	# switches	–	40	40
	# comparators	–	7	7
	# bits of the ADC2	–	–	8

number of elementary two-input logic gates needed to implement it, which depend on its number of bits (N_{AREG}), should be added to the previous result. Let us consider an IEEE1149.4 infrastructure with an MCD, $N_{ABM}=5$, $N_{DBM}=92$, $N_{DREG}=8$ and $N_{AREG}=8$, which is similar to the one used in the two provided examples (Section 5). In the present case, we only consider 92 out of the initial 100 DBMs, as 8 are now considered as part of the DCDR. The infrastructure plus the MCD has thus an equivalent complexity of $N_{(G2,SWITCHES,COMPARATORS)}=(7560,40,7)$. Table 7 resumes the equivalent complexity of the three infrastructure stages considered.

The previous examples allow concluding that the MCD represents an overhead of 33%, in relation to the digital part of the IEEE1149.4 infrastructure. In respect to the analog part, the number of switches and comparators remains the same, although one should consider the overhead associated with the ADC2 (with 8-bits, in our example).

Another important limitation of the MCD derives from its operation inside the IEEE1149.4 infrastructure. As earlier explained, the DCDR re-uses all or some of the DBMs associated with the device functional digital pins, according to the designer's debug needs. While the IC is on its normal operation mode, the DBMs may be placed in the transparent mode and thus may be used for storing the Limit_A and the Limit_B. When the IC is placed on test mode, the DBM outputs are controlled through the IEEE1149.4 infrastructure. However, the *INTEST2* and *EXTEST2* optional instructions assume the U stage of the DBMs integrating the DCDR to contain the Limit_A, which will thus be applied at the DBM outputs. This situation is not critical as the values typically used in debug operations are related to the circuit normal operation and thus will not present a particular hazard to the devices externally connected to the pins associated with the DBMs in question. Nevertheless, a DCDR supporting, at the same time, the storage of a Limit_A and the control of the DBM outputs would require 3 memory elements, thus slightly increasing the associated overhead. This limitation does not apply to the ACDR, as this is an independent register added to the IEEE1149.4 mandatory group of test data registers.

A final limitation of the MCD concerns its response time that depends on (i) the conversion time of the ADC-type selected, and (ii) the structure of the condition detector registers. In fact, the serial nature of the condition evaluation process (derived from the concatenation of a series of elementary 1-bit condition detectors) imposes a total detection time that corresponds to the sum of the individual detection time periods, which depend on the circuit implementation technology used.

7. Conclusion and future directions

The present work proposes the reuse of the IEEE1149.4 infrastructure for supporting debug operations. However, some improvements are still possible, especially concerning the effectiveness and the efficiency of the proposed solution.

The resources of the IEEE1149.4 infrastructure were extended at the ABM level so as to support basic debug operations (defined in Section 3). A further step could be done at the TBIC level, namely to allow reusing the analog test bus (AB1/2) as an internal mechanism for selecting the analog node under debug, and route it to the MCD or another internal debug mechanism. These two analog lines must either be connected to the AT1/2 pins or to the internal V_{CLAMP} voltage, as there is no configuration (for the mandatory test instructions defined in the IEEE1149.4 std.) allowing its use for debug operations such as the previously referred one.

In respect to the advanced debug operations, the most natural direction follows two lines: (i) improve the MCD functionality (e.g. reduce its time latency); and (ii) develop new debug mechanisms supporting the execution of *real time analysis* operations, which necessarily imply the memorization of circuit states (in the MS domain) until, before, or in between user-defined circuit conditions.

Other open issues include the verification of both the integrity and impact of components supporting the proposed infrastructure extensions. As a matter of fact, the PCBs that contain one or more IEEE1149.1-compatible ICs normally undergo a BST integrity test before using this test infrastructure for testing the board interconnections. This integrity test can target the TDI–TDO, TMS and TCK interconnections [44], or include the internal test logic as well [45]. An integrity test for the mandatory IEEE1149.4 infrastructure has already been described in [46], and should now be extended to include the proposed extensions and the MCD. The impact of the IEEE1149.4 infrastructure during parametric tests was already analyzed in [47,48,49], although this should now be repeated taking into account the proposed extensions.

Another aspect concerns the verification of the proposed extensions in silicon. As previously explained we chose to simulate the proposed mechanisms due to advantages such as versatility and the higher level of controllability/observability of the circuit nodes, in the time domain. These advantages are however impaired by the large simulation times. The validation involved a large number of PSpice models, for the OrCAD simulation environment, distributed in a design scheme with 39 pages (or sheets) and 8 hierarchical levels. The total simulation time, relating to the 1st example provided (Fig. 12), was of 1×10^3 s, in the following conditions: $f_{TCK}=50$ kHz; circuit simulated time$=6\times10^{-3}$ s; characteristics of the machine used=Pentium IV, 2.66 GHz, 1 GB RAM, Windows XP. This gives a relation of approximately $1{:}150\times10^3$ in terms of real time to simulated time, which is an obvious disadvantage of this sort of validation approach.

The proposed debug mechanisms reuse the IEEE1149.4 infrastructure in several aspects, mainly because it is, at present, the only standard test infrastructure for MS circuits. Yet, it is important to consider new infrastructures, e.g. the IEEE P1687 (IJTAG), which directly targets the "development of a methodology for access to embedded test and debug features, (but not the features themselves) via the IEEE 1149.1 TAP", as taken from [50]. A broader solution could also be devised in terms of an embedded macroblock, eventually in

the form of a dedicated microprocessor for supporting test (e.g. structural, parametric and functional), debug and maintenance (e.g. monitoring the IC internal temperature) operations.

Finally, for the examples described, the associated overhead is approx. 33%, in relation to the digital part of the IEEE1149.4 infrastructure (which, in its turn, usually represents a small fraction of the total circuit complexity), plus the ADC for the MCD. Considering an SOC with a microprocessor, memory, ADC and DAC converters, etc., the total overhead may be relatively smaller and easily acceptable face to the time-to-market and the total design verification costs, presently the major cost factor in AMS circuits [29].

References

[1] E.B. Eichelberger, T.W. Williams, A logic design structure for LSI testability, IEEE Transactions on Computers (1977) 462–468.

[2] IEEE Std. 1149.1, Standard test access port and boundary-scan architecture. IEEE Standards Board, October, 1993.

[3] S. Sunter, P1149.4—problem or solution for mixed-signal IC design? in: Proceedings of the International Test Conference, 1997, p. 625.

[4] R.M. Sedmak, Boundary scan: beyond production test, in: Proceedings of VLSI Test Symposium, 1994, pp. 415–420.

[5] A. Halliday, G. Young, A. Crouch, Prototype testing simplified by scannable buffers and latches, in: Proceedings of the International Test Conference, 1989, pp. 174–181.

[6] Texas Instruments, SCOPE^{TM} Logic Products: Application and Data Manual, 1994, ISBN:3-88078-098-6.

[7] National Semiconductors, System Test Access IEEE 1149 (JTAG) Solutions, 〈http://www.national.com/analog/interface/scan〉 (accessed on July 23rd, 2009).

[8] National Semiconductors, SCANSTA476—Eight Input IEEE 1149.1 Analog Voltage Monitor, 〈http://www.national.com/pf/SC/SCANSTA476.html〉 (accessed on July 23rd, 2009).

[9] J.S. Matos, A.C. Leão, J.C. Ferreira, Control and observation of analog nodes in mixed-signal boards, in: Proceedings of the International Test Conference, 1993, pp. 323–331.

[10] L. Whetsel, An IEEE 1149.1 based logic/signature analyzer in a chip, in: Proceedings of the International Test Conference, 1991, pp. 869–878.

[11] JTAG TAP Controller PP4XX Cores, IBM PowerPC Application Note, 2007, 〈http://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/C9C0D48DDA776C968725723700676D0E〉 (accessed on January 18th, 2009).

[12] IEEE-ISTO Std.5001, The Nexus 5001^{TM} Forum Standard for a Global Embedded Processor Debug Interface. IEEE Industry Standards and Technology Organization (IEEE-ISTO), December, 2003.

[13] LeCroy Corporation, 〈http://www.lecroy.com〉 (accessed on January 18th, 2009).

[14] Agilent Technologies, 〈http://www.agilent.com/〉 (accessed on January 18th, 2009).

[15] N.-C. Lee, A hierarchical analog test bus framework for testing mixed-signal integrated circuits and printed circuit boards, Journal of Electronic Testing: Theory and Applications 4 (1993) 361–368.

[16] K.-J. Lee, S.-Y. Jeng, T.-P. Lee, A new architecture for analog boundary scan, in: Proceedings of the IEEE International Symposium on Circuits and Systems, 1995, pp. 409–412.

[17] A.H. Bratt, R.J. Harvey, A.P. Dorey, A.M.D. Richardson, Design-for-test structure to facilitate test vector application with low performance loss in non-test mode, IEE Electronics Letters 29 (16) (1993) 1438–1440.

[18] IEEE Std. 1149.4. 1999, Standard for a Mixed-Signal Test Bus. IEEE Standards Board, June, 1999.

[19] R. Schuttert, D.C.L. van Geest, A. Kumar, On-chip mixed-signal test structures re-used for board test, in: Proceedings of the International Test Conference, 2004, pp. 375–383.

[20] S.K. Sunter, Cost/benefit analysis of the P1149.4 mixed-signal test bus, IEE Proceedings on Circuits, Devices and Systems 143 (6) (1996) 393–398.

[21] S.K. Sunter, B. Nadeau-Dostie, Complete, contactless I/O testing—reaching the boundary in minimizing digital IC testing cost, in: Proceedings International Test Conference, 2002, pp. 446–455.

[22] P. Syri, J. Hakkinen, M. Moilanen, IEEE 1149.4 compatible ABMs for basic RF measurements, in: Proceedings of Design Automation and Test in Europe, 2005, pp. 172–173.

[23] S.K. Sunter, Testing high frequency ADCs and DACs with a low frequency analog bus, in: Proceedings International Test Conference, 2003, pp. 228–234.

[24] S. Tatum, Lockheed Martin team integrates state-of-the-art technologies to conduct first ever demonstration of a remote controlled test and fault isolation system. Lockheed Martin Press Release, January 17th, 2002.

[25] C. Jeffrey, A. Lechner, A. Richardson, Online monitoring for automotive subsystems using 1149.4, in: Proceedings of the IEEE Board Test Workshop, 2003.

[26] SCAN STA400—National Semiconductor, 〈http://www.national.com/opf/ST/STA400EP.html〉 (datasheet accessed on January 10th, 2009).

[27] MNABST-1, Hewlett Packard Company/Matsushita Electric Industrial Co., LTD., 〈http://grouper.ieee.org/groups/1149/4/me1p.html〉 (datasheet accessed on January 10th, 2009).

[28] Cadence white paper, Cadence Company, 〈http://www.cadence.com/products/orcad/index.aspx〉 (accessed on December 10th, 2008).

[29] ITRS, International Technology Roadmap for Semiconductors. 2007 Edition, 〈http://www.itrs.net/home.html〉 (accessed on February 20th, 2008).

[30] Anadigm, 〈http://www.anadigm.com/〉 (accessed on January 10th, 2009).

[31] Lattice Semiconductor Corporation, 〈http://www.latticesemi.com/products/maturedevices/isppac/index.cfm〉 (accessed on January 10th, 2009).

[32] Cypress Semiconductor, 2007, 〈http://www.cypress.com/〉 (accessed on January 10th, 2009).

[33] IEEE Std. 1532, Standard for in-system configuration of programmable devices, IEEE-SA Standards Board, December 2001.

[34] M.G. Gericota, G.R. Alves, M.L. Silva, J.M. Ferreira, Reliability and availability in reconfigurable computing: a basis for a common solution, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16 (11) (2008) 1545–1558 November.

[35] M.C. Felgueiras, G.C. Alves, J.M.M. Ferreira, Debugging mixed-signal circuits via the IEEE1149.4 Std.—analysis of limitations and requirements, in: Proceedings of the 12th IEEE International Mixed-Signal Testing Workshop, 2006, pp. 2–7.

[36] IEEE 1149.4, Working Group Meeting Minutes, 2003.

[37] A. Gorodetsky, Bridge for on-board and on-chip 1149.4—compliant testability, in: Proceedings of the IEEE Board Test Workshop, 2005.

[38] G.R. Alves, J.M.M. Ferreira, From Design-for-test to design-for-debug-and-test: analysis of requirements and limitations for 1149.1, in: Proceedings of VLSI Test Symposium, 1999.

[39] G.O. Ducoudray-Acevedo, J. Ramírez-Angulo, Innovative built-in self-test schemes for on-chip diagnosis, compliant with the IEEE1149.4 mixed-signal test bus standard, Journal of Electronic Testing: Theory and Applications 19 (2003) 21–28.

[40] M. Slamani, B. Kaminska, T-BIST: a built-in self-test for analog circuits based on parameter translation, in: Proceedings of the Second Asian Test Symposium, 1993, pp. 172–177.

[41] M.C. Felgueiras, G.C. Alves, J.M.M. Ferreira, A built-in debugger for 1149.4 circuits, in: Proceedings of the 13th IEEE International Mixed-Signal Testing Workshop, 2007, pp. 93–98.

[42] A.V. Fidalgo, R.J. Costa, J.M. Ferreira, G.R. Alves, Experimenting the 1149.1 and 1149.4 test infrastructures in a Web-accessible remote Lab (without Plug-ins!), in: Proceedings of the XVI Conference on Design of Circuits and Integrated Systems, 2001.

[43] M.C. Felgueiras, G.C. Alves, J.M.M. Ferreira, Debugging mixed-signals circuits via IEEE1149.4—a built-in mixed condition detector, in: Proceedings of the XXII Conference on Design of Circuits and Integrated Systems, 2007.

[44] F. Jong, F. Heyden, Testing the integrity of the boundary scan test infrastructure, in: Proceedings of the International Test Conference, 1991, pp. 105–112.

[45] A.T. Dahbura, M.U. Uyar, C.W. Yau, An optimal test sequence for the JTAG/IEEEP1149.1 test access port controller, in: Proceedings of the International Test Conference, 1989, pp. 55–62.

[46] M.C. Felgueiras, G.C. Alves, J.M.M. Ferreira, Integrity checking of 1149.4 extensions to 1149.1, in: Proceedings of the XXI Conference on Design of Circuits and Integrated Systems, 2006.

[47] Duzevik, Preliminary Results of Passive Component Measurement Methods Using an IEEE 1149.4, in: Proceedings Board Test Workshop, 2002.

[48] T. Saikkonen, M. Moilanen, Component value calculations and characterization for measurements in 1149.4 environment, in: Proceedings of the 12th IEEE International Mixed-Signal Testing Workshop, 2006, pp. 76–82.

[49] M.C. Felgueiras, G.C. Alves, J.M.M. Ferreira, Measurements in 1149.4 environments—correcting the infrastructure switches influence, in: Proceedings of the IEEE Board Test Workshop, 2007.

[50] IEEE P1687 (IJTAG) Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device, 〈http://grouper.ieee.org/groups/1687/〉 (accessed on July 23rd, 2009).