An On-chip IEEE 1687 Network Controller for Reliability and Functional Safety Management of System-on-Chips

Ahmed M. Y. Ibrahim and Hans G. Kerkhoff

Testable Design and Test of Integrated Systems Group (TDT),
Centre of Telematics and Information Technology (CTIT), University of Twente,
Enschede, the Netherlands
a.m.y.ibrahim@utwente.nl and h.g.kerkhoff@utwente.nl

Abstract—The IEEE 1687 standard defines a standardized mechanism for the off-chip access of embedded instruments. A subset of these instruments are also used for maintaining the reliability and functional safety of the chip during its lifetime. For example, temperature sensors, voltage monitors and Built-In-Self-Test engines. In this paper, we present a novel on-chip controller for IEEE 1687 networks which can execute instrument procedures documented in the IEEE 1687 PDL language. These procedures are incorporated within the reliability and functional safety embedded software that uses the measurements data of the instruments. The controller includes an efficient structural model of the IEEE 1687 network and can perform on-chip pattern retargeting on arbitrary networks. In addition, it can perform localization of instrument interrupts that are propagated via multi-mode IEEE 1687 networks.

Keywords—IEEE 1687, IJTAG, embedded instruments, dependability, reliability, functional safety.

I. INTRODUCTION

A growing number of non-functional components, referred to as Embedded Instruments (EIs), is being integrated in modern System-on-Chips (SoCs) for testing, debugging and other functions. During the SoC lifetime, a subset of these EIs provides online measurements of environmental and performance parameters, performs online tests, and detects logical and physical faults across the chip.

SoCs are being increasingly used in safety- and mission-critical domains such as automotive, medical and space. Such applications require a dependable operation along with a prolonged lifetime, as in the case of Advanced Driving Assistance Systems (ADAS) [1]. Consequently, several dedicated software solutions have been previously proposed for maintaining the Reliability and functional Safety (RaS) of SoCs using EIs. For instance, on-line logic Built-in-self-test (LBIST) is a popular functional safety procedure that uses BIST engine EIs. Table I lists examples of RaS procedures.

Due to the rapid increase of the on-chip EI count, the IEEE 1687 standard (IJTAG) [2] was introduced to standardize their off-chip access methods which was conventionally done in an ad hoc manner using the IEEE 1149.1 Test Access Port (TAP).

Table I: Examples of RaS procedures and the utilized EIs.

Procedure	Utilized EIs ¹	Ref.
Dynamic Reliability Management (DRM)	T, V, PC, PLL, VR	[3]
Dynamic Thermal Management (DTM)	T, (PLL or VR or CGC)	[4]
Dynamic Power Management (DPM)	PC, I_{DD} , PLL, VR	[5]
Guardband Management	CPM, PLL, VR	[6]
Online Built-In-Self-Test	CTW, BIST	[7]
Fault Management	FD	[8]

¹T: Temperature sensor, V: Voltage monitor, PC: Performance Counter, PLL: Phase-Locked Loop, VR: Voltage Regulator, CGC: Clock Gating Circuit, I_{DD} : Current sensor, CPM: Critical Path Monitor, CTW: Core Test Wrapper, BIST: Built-In-Self-Test engine, FD: Fault Detectors.

The IEEE 1687 standard enables the ease of EIs integration, portability and reuse.

On the other hand, conventional on-chip access mechanisms to EIs range from direct connections for a small number of EIs [9], reusing the functional network [10] or using a dedicated EIs network [11]. With the increasing reliability issues in nano-technologies, future SoCs will be required to periodically sense and accordingly adapt during runtime [12]. Therefore, it is expected that RaS procedures will become more complex, and utilize more heterogeneous EIs. Consequently, accessing and controlling the EIs in those procedures using the conventional methods would become more complex to implement and will hinder their scalability and reusability.

Design scalability and reusability are key drivers of the semiconductor industry's growth, which is usually restricted by costs and time-to-market requirements. The IEEE 1687 standard introduces a simple, yet scalable network infrastructure for connecting heterogeneous EIs. As a result, reusing IEEE 1687 networks as EI networks for RaS procedures becomes a cost-efficient method for the on-chip EIs access.

In this paper, we present the analysis and design of a novel on-chip IEEE 1687 network controller. This controller is able to execute high-level RaS procedures that are implemented with no regard to the instruments access mechanism, access procedures and their physical locations. This is achieved by

Figure 1: (a) An IJTAG network with its corresponding H-Array and (b) the resulting configuration for state $[C2, C1] = [1, 0]$.

utilizing both the IEEE 1687 Procedural Description Language (PDL) in the RaS procedures development, and the on-chip pattern retargeting process.

The remainder of the paper is organized as follows: section II discusses our prior work, then a preliminary analysis is given in section III. In section IV the software flow of RaS procedures is presented, while the hardware architecture of the IJTAG network controller is presented in section V. Experimental results are presented and discussed in section VI, and finally conclusions are presented in section VII.

II. PRIOR WORK

This section discusses our prior work on IEEE 1687 network (IJTAG) structural modelling and the network architectural modification to enable efficient EI interrupts management.

A. IEEE 1687 Network Structural Modelling

Performing on-chip network access operations requires maintaining a comprehensive model of the IJTAG network. Since the IEEE 1687 standard presents a large design space of flexible network organizations, the model should be formally derived according to the standard specifications.

IJTAG networks are based on reconfigurable scan [13]. State-of-the-art IJTAG network models are mostly graph-based which only capture the components connectivity, (e.g. [13] and [14]). Since resolving the inter-registers structural and temporal dependencies is required for network operations, using graph-based models for modeling the network would require run-time resolution of the dependencies which can be a complex task for arbitrary networks. Therefore, we proposed a linear network model that embeds both the connectivity as well as the dependency information for a more efficient on-chip access [15]. The model, referred to as the Hierarchical Array (H-Array), can be formally constructed for arbitrary IJTAG networks via the procedure detailed in [15]. Figure 1(a) shows an IJTAG network and its corresponding H-Array.

The H-Array consists of several nested *dependency sections* identified by their header elements (indicated in Figure 1 by the coloured section indicators to the right). A dependency section represents a structural dependency between the enclosed

Figure 2: The Extended SIB.

elements and the corresponding control bit (or a combinational function of several ones) which is referenced in the header.

A dependency section is said to be active if the control bit referenced in the header element has an updated value in the network that corresponds to the activation condition of such a section. For instance, the section indicated by the header (H_{I1_1}) is active when $C1 = '1'$. Traversing the H-Array while skipping the inactive sections will result in traversing a sequence of register elements that are equivalent to the sequence of registers forming the active scan path. Figure 1(b) shows the network state $[C2=1, C1=0]$ (top), and the corresponding state of the H-Array (bottom) where the inactive sections are made blank. It can be shown that traversing the active array elements of type “scan register” in the H-Array is equivalent to the active scan path from TDO to TDI [15].

B. Multi-Mode IEEE 1687 Networks

IJTAG networks are of a master-slave type, where the network controller is the master and the EIs are slaves, and hence are not originally specified to support delivering an EI interrupt. Supporting interrupts delivery using hierarchical IJTAG networks was presented in [16]. We enhanced this asynchronous delivery mechanism to provide a scalable EIs interrupts delivery and efficient localization using the hierarchical multi-mode IJTAG networks concept in [17].

Hierarchical IJTAG networks are based on the Segment Insertion Bit (SIB) component, where a SIB allows to include and exclude an attached scan segment. Multi-mode networks extend SIBs for including in-line bypassable interrupt flag registers (figure 2(a)). Figure 2(b) shows an example hierarchical network and Figure 2(c) shows the resulting asynchronous interrupt propagation network based on the modular Extended SIB (ESIB) design.

The in-line flags are included or excluded according to the network mode. Three modes were defined: A) normal access (no flags in the scan path), B) diagnosis access (flags are included) and C) on-chip localization (only the flags in the interrupting instrument hierarchical access path are included).

III. PRELIMINARY ANALYSIS

Modern technology nodes introduce increasing variations and complex failure mechanisms. Consequently, the RaS pro-

Figure 3: An IEEE 1687 based clustered design layer of a SoC for executing reliability and functional safety applications.

cedures are expected to become more processing intensive [12]. As a result, decoupling the functional processing from the RaS one becomes required to optimize the design space for RaS processing, as well as to minimize the impact of the RaS processing on the functional one. Utilizing IJTAG networks for RaS communication is in-line with this requirement. Furthermore, a dedicated processing unit for executing RaS procedures is adopted to realize this decoupling. The processing unit should also be able to access and control the IJTAG networks for EIs access; we refer to this unit as the Dependability Manager (DM).

Hierarchical RaS management has been shown to enhance the design scalability and reduce its complexity [18]. Since IJTAG networks can be physically organized in a hierarchical manner, this can be leveraged to enable hierarchical RaS management by inserting a DM to control a sub-network. This sub-network connects the EIs in a certain cluster of the chip where its reliability and functional safety can be managed individually (Figure 3). A system DM manages the reliability and functional safety of the entire chip and communicates with the cluster DMs via the IJTAG network [19].

Furthermore, a typical RaS procedure has three main phases: 1) sensing, 2) processing and 3) actuation, that are periodically performed during the lifetime of the SoC. While processing is done by the DMs, physical sensing and actuation are performed by the IJTAG-integrated EIs, where their operating procedures are documented in PDL. An EI provider usually supplies an IJTAG-wrapped EI along with its PDL operating procedures. Given the increasing number of heterogeneous EIs, and since the EIs operating procedures are already becoming complex due to the increasing complexity of the EIs themselves, using the PDLs in the development of the RaS procedures will facilitate their scalable and reusable implementation.

In order to reuse the PDLs in different networks, the IEEE

```

1 void DRM (void) { /*periodic procedure */
2   /* calls to functions defined in header files
   to be linked with the compiled PDL iProcs*/
3   T[t] = ReadTemp();
4   V[t] = MeasureVolt();
5   Rel[t] = Calc_reliability(T, V, t);
6   If (Rel[t] < 0.99){
7     .....
8   }
9   t++;
10 }

```

(a) Embedded C/C++ RaS procedure

```

1 iProcsForModule Temp_Sensor
2 iProc ReadTemp {
3   iWrite en 0b1;
4   iApply;
5   iRunLoop 1000 -tck
6   iRead Temp;
7   iWrite en 0b0;
8   iApply;
9   set T [iGetReadData Temp]
10  return $T
}

```

(b) Temperature instrument PDL procedure for measurements

Figure 4: Integration of PDL in high-level RaS procedure.

1687 standard defines a process referred to as *retargeting* which translates the PDL access commands that are written at the EI-level, to network-level scan vectors. Retargeting can be performed offline; then the resulting vectors are shifted in runtime, however this is only suitable for static procedures in PDL-0 [2]. For dynamic EI procedures where their control flow is dependent on the runtime EIs data (i.e. PDL-1), dynamic retargeting should be performed. Therefore, we support dynamic retargeting in the on-chip execution of the RaS procedures using a dedicated retargeting engine.

IV. CROSS-COMPILATION OF RAS PROCEDURES ALONG WITH INSTRUMENT PDLs

An RaS procedure could either be *periodic* or *interrupt-driven*, where the interrupt source is usually an EI, and is handled by a specific Interrupt Service Routine (ISR). For example, the periodic estimation and profiling of the reliabilities of the processing modules in [3] is a periodic procedure, while for instance in fault management [8], the procedure is activated only in case a fault is detected by one of the instruments, and is therefore an interrupt-driven procedure.

In this work, a programming model of the RaS procedures is introduced which allows them to be provided as a library of reusable procedures. In this model, periodic and event-triggered RaS procedures are written in a high-level programming language (e.g. C/C++) in a generic manner, and with no regard to the instrument access procedures or the instruments location in the IJTAG network. The procedures include instrument-access function calls which are linked to the PDL procedures (iProc). Figure 4 (a) shows an example of a high-level RaS procedure including calls to PDLs, and in Figure 4 (b) an example is shown of a simple PDL-0 procedure to operate a temperature sensor.

PDL is an extension of the TCL scripting language [2], and is consequently not intended for embedded applications and is rather executed using interpreters. As a result, a methodology for porting the PDL procedures to an embedded processing environment has been developed. Embedded porting of TCL

Figure 5: Cross-compilation flow of RaS procedures.

commands can be performed in a similar manner as in the embedded C language. To achieve this, a categorization of the 24 different PDL commands [2] into four different categories is being proposed: 1) non relevant commands (e.g. iNote), 2) relevant to the compiler but not to the embedded processing (e.g. iProcsForModule), 3) compiled to native processor instructions (e.g. iGetMiscompares) and 4) compiled to retargeting engine instructions (e.g. iWrite).

Figure 5 shows the compilation flow of the RaS procedures. Both the periodic procedures and the ISRs are compiled using a high-level language cross-compiler for the target processing unit. References to iProcs are left during the compilation to be linked to the compiled PDLs. In addition, the references to the EIs registers that are included in the PDL access commands (iWrite, iRead and iScan) are linked to their corresponding index in the network H-Array (as illustrated in Figure 6). The interrupting EI index in the H-Array is provided as a directive to the compiler in the beginning of the ISR, such that the ISR locations in the instruction memory can be generated and provided as an Interrupt Vector Table (IVT).

V. ARCHITECTURE OF THE IEEE 1687 NETWORK CONTROLLER

A dedicated processing unit for the RaS procedures (i.e. the DM) is being proposed in order to achieve a functionally-decoupled RaS processing. The DM executes the RaS procedures and performs the network-access operations (e.g. retargeting and interrupts localization).

A. Architectural Overview

The DM (Figure 7) consists of a central processor unit (including instruction and data caches), a set of co-processors for network access and a set of peripherals and memories. The processor unit is considered as an Application Specific Instruction-set Processor (ASIP) dedicated for RaS procedures [20].

The DM incorporates three co-processors for network access: 1) a retargeting engine, 2) an Interrupt Management Unit (IMU) and 3) a reconfigurable Built-In-Self-Test (BIST) engine. An IEEE 1149.1 controller that implements the Capture-

Figure 6: (a) A wrapped temperature sensor and (b) its corresponding H-Array section and (c) the linker mapping file.

Shift-Update (CSU) cycle [2] is included in order to generate the IJTAG control signals with the correct timings according to the standard. The DM accesses the network via the IJTAG host interface only if no external controller is connected [17]. Since the processor unit schedules the operations of the co-processors, it also controls which co-processor can access the network via the 1149.1 controller at a certain time using a control switch.

The BIST engine can perform core-based testing on IEEE 1500 wrapped cores. In this case the wrappers are considered to be EIs accessed by the retargeting engine, where the written data to the boundary registers and scan chains (i.e. core patterns) are generated by the BIST engine [21].

A ROM for the H-Array is included and accessed by both the retargeting engine and the IMU. Another ROM is also included for IVT, which implements a mapping between the H-Array index of the localized interrupting EI with the location of the corresponding ISR in the instruction cache. In addition, a timers unit is included in order to schedule the periodic procedures by issuing an internal interrupt to the processor in the case a procedure is due. Then using the IVT, the location of the periodic procedure in the instruction cache can be located.

Finally, the DM has two communications ports, the first is the IJTAG client interface that is connected to a set of TDRs that are accessed by an off-chip IJTAG controller for coordinating the network control and for quick localization [17]. In addition, the off-chip controller could reuse the BIST engine for e.g. production testing by directly controlling the BIST

Figure 7: Architecture of the Dependability Manager.

instruments via the TDRs [22]. The second communication port is an interface for the functional network which allows for performing cross-layer RaS procedures [19].

B. The Retargeting Engine

The retargeting engine co-processor translates the EIs access commands (resulting from the compilation of iRead, iWrite and iScan) to the scan vectors required to configure the network and access the EIs. Consequently, the retargeting engine allows a generic implementation of the RaS procedures regardless of the network topology or the locations of the EIs. It receives a set of register IDs (corresponding to their locations in the H-Array) from the processor unit to be concurrently accessed, with the write values for the write instructions. It subsequently generates the required scan vectors, and then provides the read values for the read instructions.

The retargeting engine implements a dynamic retargeting methodology using the H-Array as a comprehensive network model. Such methodology is referred to as *structured retargeting* [23], [24]. Structured retargeting is a light-weight yet comprehensive dynamic retargeting method that can be efficiently implemented on-chip.

C. The Interrupt Management Unit

The IMU is responsible for handling the received asynchronous interrupts from the multi-mode IJTAG network. An interrupt could either be preemptive (PI) (e.g. critical faults) or non-preemptive (NPI) (e.g. instruments operation status) [17]. In case of a PI, the IMU stops the operation of the processor unit and saves its registers (including the program counter) to an internal set of temporary registers; then it performs a hierarchy-aware localization using the H-Array. When the interrupt source is localized, the corresponding ISR address in the instruction cache is fetched from the IVT, and the processing unit could start immediately servicing the interrupt. In case of an NPI, the processing unit is allowed to finish the on-going periodic procedure before servicing the interrupt.

In order to be able to perform localization on multi-mode networks, a dedicated array element type for the Extended SIBs (ESIB) is introduced to the H-Array. Figure 8(a) shows

Figure 8: Hierarchy-aware interrupt localization for I2.

the corresponding H-Array section of the ESIB that is inferred during normal network access.

The hierarchy-aware localization is performed by first setting the network to the localization mode by asserting a dedicated control signal (Loc). Asserting the ‘Loc’ signal immediately configures a scan path between the TDI and TDO ports of the host interface, that is formed of only the flag register in the ESIBs located on the minimum hierarchical access path to the interrupting EI. The IMU applies one CSU cycle using the 1149.1 controller. During the shift cycle, it traverses the H-Array according to the shifted-out bits.

The traversal starts with the first ESIB entry in the H-Array. If a ‘0’ is received, the traversal skips the corresponding ESIB section, while if a ‘1’ is received, the traversal steps inside the ESIB section. This process is repeated until a TDR entry is encountered while stepping into a section. The corresponding H-Array location of this TDR becomes the ID of the interrupt in the IVT. Figure 8(b) shows the corresponding H-Array to the network in Figure 2(b) and the localization steps for the interrupting EI “I2”.

VI. EXPERIMENTAL SETUP AND RESULTS

An example realization of the DM and its supporting cross-compilation flow has been developed. We have chosen a 32-bit MIPS processor as the central processor unit in the DM (Figure 7), since MIPS is a simple processor with a rich set of software tools. MIPS defines a set of co-processor instructions that allow data movements to/from the co-processors. The MIPS co-processor instructions were extended to implement specialized ones for the retargeting engine. The retargeting engine architecture has been described in [15] and then modified to implement structured retargeting [24]. A wrapper was further designed for integrating it as a MIPS co-processor.

A PDL cross-compiler was developed for MIPS using the ANTLR tool [25]. The grammar of both the PDL commands and a subset of the main TCL commands was developed and used to construct the cross-compiler. The compiler follows the PDL embedded porting guidelines described in section IV.

We used this setup to evaluate the efficiency and scalability of the proposed IJTAG controller for the execution of RaS procedures in terms of the overall registers access time (OAT). For dynamic retargeting the following holds: $OAT = retargeting\ time + network\ access\ time$.

Table II: Evaluation Results

# TDRs	# HL		# SIBs		H-Array Size (DxW bits)		OAT			
	BT	QT	BT	QT	BT	QT	BT		QT	
							Rst	Cnf	Rst	Cnf
25	6	3	49	31	123x11	87x11	169	48	103	48
50	7	4	99	63	248x12	176x12	216	54	165	63
100	8	4	199	125	498x13	350x13	269	60	165	63
200	9	5	399	250	998x14	700x14	328	66	242	78
500	10	5	999	625	2498x16	1750x15	393	72	242	78
1000	11	6	1999	1250	4998x17	3500x16	464	78	334	93

A set of balanced rooted tree networks have been considered for this evaluation, with the TDRs of the EIs at the leaves. The Binary Trees (BT) are considered as an example of a network with deep dependencies, and Quinary Trees (QT) (with five children per SIB) for a more balanced SIBs to TDRs ratio.

For each network, the corresponding H-Array was generated. A PDL file has been developed with two iApply groups of an access command (iRead) to a TDR. The first corresponds to an access after Reset (Rst), and the second corresponds to a repeated access after network configuration (Cnf). The compiled PDL is loaded into the instruction cache of the DM, and using RTL-level simulation, the OAT for this PDL is reported.

Table II shows the results for BT and QT trees connecting an increasing number of TDRs. The number of Hierarchical Levels (HL) and number of SIBs for each network are shown in columns 2, 3 and 4, 5 respectively. Columns 6 and 7 show the size of the generated H-Arrays, while columns 8-11 report the OAT for each tree type and for an Rst and Cnf access.

The OAT results show that while the number of TDRs was multiplied by 40 (i.e. from 25 to 1000), the OAT was only multiplied by 2.7 for a BT with Rst access and 1.63 for Cnf access. For QT it was 3.2 times for Rst and 1.9 for Cnf. This indicates that using IJTAG for the on-chip access to EIs via the proposed DM, the OAT for an EI could be efficiently scaled for large number of EIs which enables the scalability and reuse of the RaS procedures.

VII. CONCLUSIONS

Software procedures for managing the reliability and functional safety of SoCs are expected to become increasingly complex and utilize a growing number of heterogeneous EIs due to the increased SoC complexities and the decreased reliability of nanotechnologies. In this work, the execution of those procedures using IJTAG networks has been introduced as a scalable, reusable and cost-efficient solution. PDLs would be an integrated part of the procedures to enable their implementation in a reusable manner with no regard to the EIs location and operating procedures. An IJTAG network controller has been presented and validated along with its software tool flow. The proposed solution was shown to enable a scalable and reusable implementation of RaS procedures.

VIII. ACKNOWLEDGEMENT

This research was carried out within the EU-PENTA project "HADES", financed by the European Commission (EC) and the Netherlands Enterprise Agency (RVO). Also the authors

would like to thank M. Zakiy for his work on the DM implementation and tool-flow development.

REFERENCES

- [1] U. Backhausen et al., "Robustness in automotive electronics: An industrial overview of major concerns," International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 157-162, 2017.
- [2] IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device, IEEE Std 1687-2014, 2014.
- [3] C. Zhuo, D. Sylvester and D. Blaauw, "Process variation and temperature-aware reliability management," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 580-585, 2010.
- [4] K. Skadron, et al., "Temperature-aware microarchitecture," Int'l Symposium on Computer Architecture (ISCA), pp. 2-13, 2003.
- [5] Canturk Isci, et al., "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget," Int'l Symposium on Microarchitecture (MICRO), pp. 347-358, 2006.
- [6] C. R. Lefurgy et al., "Active Guardband Management in Power7+ to Save Energy and Maintain Reliability," IEEE Micro, vol. 33, no. 4, pp. 35-45, 2013.
- [7] H. Inoue, Y. Li and S. Mitra, "VAST: Virtualization-Assisted Concurrent Autonomous Self-Test," Int'l Test Conf. (ITC), pp. 1-10, 2008.
- [8] K. Shibin, S. Devadze and A. Jutman, "On-line fault classification and handling in IEEE1687 based fault management system for complex SoCs," Latin-American Test Symposium (LATS), 2016, pp. 69-74.
- [9] M. Saen et al., "Embedded SoC Resource Manager to Control Temperature and Data Bandwidth," IEEE International Solid-State Circuits Conference. Digest of Technical Papers, 2007, pp. 296-604.
- [10] S. Velusamy et al., "Monitoring temperature in FPGA based SoCs," International Conference on Computer Design (ICCD), pp. 634-637, 2005.
- [11] S. Madduri et al., "A monitor interconnect and support subsystem for multicore processors," Design, Automation & Test in Europe (DATE), pp. 761-766, 2009.
- [12] J. Henkel et al., "Reliable on-chip systems in the nano-era: Lessons learnt and future trends," Design Automation Conference (DAC), pp. 1-10, 2013.
- [13] R. Baranowski, M. A. Kochte and H. J. Wunderlich, "Scan pattern retargeting and merging with reduced access time," IEEE European Test Symposium (ETS), pp. 1-7, 2013.
- [14] F. Ghani Zadegan et al., "Design, Verification and Application of IEEE 1687," Asian Test Symposium (ATS), pp.93-100, 2014.
- [15] A. Ibrahim and H. G. Kerkhoff, "Analysis and design of an on-chip retargeting engine for IEEE 1687 networks," European Test Symposium (ETS), pp. 1-6, 2016.
- [16] A. Jutman, S. Devadze and K. Shibin, "Effective Scalable IEEE 1687 Instrumentation Network for Fault Management," IEEE Design & Test, vol. 30, no. 5, pp. 26-35, Oct. 2013.
- [17] A. Ibrahim and H. G. Kerkhoff, "Efficient utilization of hierarchical IJTAG networks for interrupts management," Int'l Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 97-102, 2016.
- [18] Liang Guang et al., "Hierarchical agent monitoring design approach towards self-aware parallel systems-on-chip," ACM Transactions on Embedded Computing Systems (TECS), vol. 9, no. 3, pp. 124, Feb 2010.
- [19] A. Ibrahim and H. G. Kerkhoff, "Cost-Efficient Dependability Management Framework for Self-aware System-on-Chips based on IEEE 1687," Int'l Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 1-2, 2017.
- [20] A. Nohl, F. Schirmeister and D. Taussig, "Application specific processor design: Architectures, design methods and tools," Int'l Conference on Computer-Aided Design (ICCAD), pp. 349-352, 2010.
- [21] A. Ibrahim and H. G. Kerkhoff, "Towards an Automated and Reusable In-Field Self-Test Solution for MPSoCs," Int'l Conference on Microelectronics (ICM), pp. 1-6, 2016.
- [22] A. Ibrahim and H.G. Kerkhoff, "IJTAG integration of complex digital embedded instruments," Int'l Design & Test Symposium (IDT), pp.18-23, 2014.
- [23] A. Ibrahim and H. G. Kerkhoff, "Structured scan patterns retargeting for dynamic instruments access," IEEE VLSI Test Symposium (VTS), pp. 1-6, 2017.
- [24] A. M. Y. Ibrahim, A. Ibrahim, M.Safar, M. Watheq El-Karashi and H. G. Kerkhoff, "Efficient structured scan patterns retargeting for hierarchical IEEE 1687 networks," IEEE VLSI Test Symposium (VTS), pp. 1-6, 2019.
- [25] ANTLR, "ANother Tool for Language Recognition", [Online]. Available: www.antlr.org/