Towards Design And Validation Of Mixed-Technology SOCs

S. Mir[1], B. Charlot[1], G. Nicolescu[1], P. Coste[1], F. Parrain[1], N. Zergainoh[1],
B. Courtois[1], A. Jerraya[1] and M. Rencz[2]
[1]**TIMA Laboratory, 46 Av. Félix Viallet, 38031 Grenoble, France**
[2]**Technical Uni. Budapest, Goldmann Gyorgy tèr 3, 1521 Budapest, Hungary**

Abstract

This paper illustrates an approach to design and validation of heterogeneous systems. The emphasis is placed on devices which incorporate MEMS parts in either a single mixed-technology (CMOS + micromachining) SOC device, or alternatively as a hybrid system with the MEMS part in a separate chip. The design flow is general, and it is illustrated for the case of applications embedding CMOS sensors. In particular, applications based on fingerprint recognition are considered since a rich variety of sensors and data processing algorithms can be considered. A high level multi-language/multi-engine approach is used for system specification and co-simulation. This also allows for an initial high-level architecture exploration, according to performance and cost requirements imposed by the target application. Thermal simulation of the overall device, including packaging, is also considered since this can have a significant impact in sensor performance. From the selected system specification, the actual architecture is finally generated via a multi-language co-design approach which can result in both hardware and software parts. The hardware parts are composed of available IP cores. For the case of a single chip implementation, the most important issue of embedded-core-based testing is briefly considered, and current techniques are adapted for testing the embedded cores in the SOC devices discussed.

Index terms: Design, Verification, SOCs, MEMS, HDLs, Co-simulation, Architecture exploration.

1 Introduction

The integration of a complete system on a single chip or SOC (System-On-a-Chip) poses many new challenges to design and test engineers. They are truly heterogeneous systems mixing hardware/-software and digital/analog parts. SOCs already embed typical subsystems such as DSP, RAM, ROM, MPEG cores, etc ... and they may soon include MEMS (Micro-Electro-Mechanical Systems) or microsystem cores [1].

The specification of SOC devices includes hardware and software. This can be either homogeneous, i.e. using a single language for the specification of the whole system, or heterogeneous, i.e. using specific languages for different cores [2]. This paper illustrates one possible way of addressing this by means of

- a multi-language/multi-engine simulation for high level system design and validation, and
- one single-language/single-engine simulation for low-level design and validation of a sub-system.

The overall high level design problem is decomposed into lower level design sub-problems. This approach is illustrated in this paper for the case of applications based on CMOS sensors, in particular those requiring fingerprint recognition. These applications are based on obtaining an image of a fingerprint via an array of sensor pixels. The information obtained from the sensor is processed by the system and compared with a database of fingerprints to provide a go/no-go recognition signal. Flexibility exists in the design of such applications, in terms of the different types of sensors that can be used and the algorithms required for data processing. The system architecture must then be optimized with respect to critical application dependent variables such as the level of security required, the speed of the system to deliver a go/no-go recognition response, and the cost of the implementation in terms of the surface and Intellectual Property (IP) cores used. The high level multi-language/multi-engine approach used for system specification and co-simulation facilitates the task of an early system architecture exploration.

A SOC architecture for the required application is next automatically generated via a co-design approach. The type of architectures generated is quite simple and flexible, being well suited for the requirements of the types of applications based on fingerprint recognition. For example, a change in the type of sensor, which may imply the use of additional or different data processing algorithms, may require the use of different IP cores (e.g. a different CPU or different memory cores) according to the security, speed and cost requirements of the application. The cores may change in type and quantity, but the type of architecture automatically generated is still the same.

Conceptually, designing a SOC may appear analagous to the design of a printed-circuit board, but this analogy disappears once the test/validation and diagnostic/debugging of the SOC are taken into account [3]. This is a main concern for embedding MEMS cores, since there is at present little knowledge about testing these parts affected by new types of fabrication defects and failure mechanisms [4]. Current work related to embedded-core-based testing is briefly considered at the end of the paper and its implication for testing the embedded MEMS core discussed.

2 High-Level Heterogeneous System Design and Validation

To date, most of the existing co-design systems are based on a single paradigm or language. Languages such as SDL, Statechart, C, VHDL or C++ are used for the specification of both hardware and software. But experiences with formal specification languages have shown that there is no unique universal specification language for all kinds of applications. The use of specification languages has to be selectively targeted. Some of these languages are more suitable for state-based specification (e.g. SDL or Statechart). Some others are more suited for data computation (LUSTRE, SILAGE) while many others are more suitable for algorithmic specifications (e.g. C or ADA). Moreover, in practice, a large system can be designed by separate groups, which may have different background tools and expertise and use different modeling styles. Besides, multilanguage specification is driven by needs of modular and evolutive design due to increasing levels of complexity. Modurality helps in mastering this complexity by promoting design reuse and, more generally, encouraging concurrent engineering. Unless a single unique SLDL (System Level Design Language) is imposed, a multi-language approach appears to be the way forward [5].

This is illustrated here by means of the system of Figure 1 representing a general architecture of a communications device. The device is made of four heterogeneous sub-systems that are traditionally designed by separate groups that may be geographically distributed.

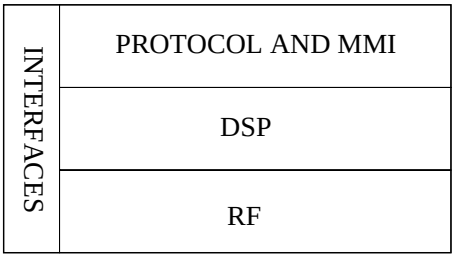

Figure 1: General architecture of a communications device.

The protocol and MMI sub-system is in charge of high-level protocols and data processing and user interface. It is generally designed by a software group using high-level languages such as SDL or C++. The DSP sub-system is in charge of signal processing and error correction. It is generally designed by a DSP group using specific tools and methods such as Matlab/Simulink or COSSAP. The radio-frequency (RF) sub-system is in charge of the physical connection. It is generally made by an analog design group using another kind of specific tools and methods such as CMS. It may also include MEMS sub-parts.Finally, the interface sub-system is in charge of the communication between the three other parts. It may include complex busses and a sophisticated memory system. It is generally designed by a hardware group using classical EDA tools.

The key issue for design automation of such a system is the validation of the overall design and the synthesis of the interfaces between the different sub-systems. Validation of the overall system can be done using the co-simulation based-approach. Of course, most of these sub-systems may include both hardware and software. The use of a multilanguage specification requires techniques able to handle a multiparadigm model. Instead of simulation we will ned co-simulation and instead of verification we will need co-verification. Additionaly, multilanguage specification brings about the issue of interfacing subsystems which are described in different languages. The interfaces need to be refined when the initial specification is mapped onto a prototype.

Figure 2 shows a generic flow for co-design starting from multi-language specification. Each sub-system of the initial specification is described in a specific language and may need to be decomposed into hardware and software parts. Moreover, we may need to compose some of these sub-systems in order to perform global hardware/software partitioning. In other words, partitioning may be local to a given sub-system or global to several sub-systems. The co-design process also needs to tackle the refinement of interfaces and communication between sub-systems. From the system level down to the implementation level, the synthesis of the protocols and the generation of the required interfaces is automatically achieved. A mixed hardware and software implementation for the communication between components is added. Finally, the co-design approach must also take into account the needs for testing the highly embedded system.

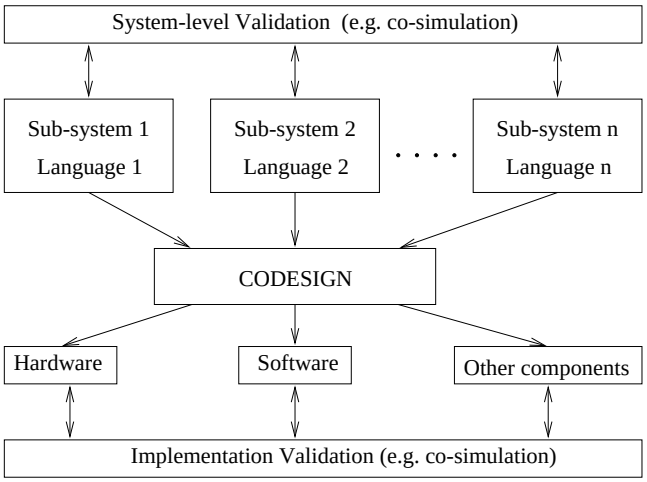

Figure 2: Multi-language co-design.

3 Mixed-technology SOCs for applications based on fingerprint recognition

Devices based on biometric recognition, i.e. the identification of people through unique physical characteristics, and in particular fingerprint recognition [6, 7], are recently finding market in numerous applications requiring fast, low cost and secure identification. Fingerprint recognition can replace passwords, PINs, keys, cards and badge systems with the convenience that one can not loose, forget or steal it. Biometric systems can find applications in the control of access to houses, offices, cars, airports and so on. And also for database network access like workstation login, intranets, online transactions, banking and e-commerce. Until now biometric authentification requires complex capture systems coupled with communication devices and computers for image processing and signature matching.

The integration on the same chip (SOC) of a sensing unit and its control and computing electronics allows lowering cost and size and offers many more additional applications. Highly integrated fingerprint recognition systems can take advantage of its small dimensions and low power consumption, being then very interesting for emerging hand held devices such as Personal Digital Assistants (PDAs), organisers, cellular phones, mobile computers or smart cards.

Such SOCs aimed at fingerprint recognition are truly heterogeneous systems, with digital, analog, mixed-signal and MEMS parts which are better specified using different languages.

3.1 Types of sensors and their architecture

It is possible to classify fingerprint sensors in different ways, as exemplified in Table 1. Sensors can be classified according to the energy domain in which the read mechanism works. Thus the sensor may employ, for example, mechanical, optical, electrostatic or thermal physical principles for acquiring the data related to the fingerprint. The sensor is made of an array of pixels, each pixel covering a small part of the fingerprint. A pixel senses the differences between ridges and valleys in the finger in terms of differences of pressure, distance, capacitance (air gap or direct contact) or temperature and heat conduction.

According to the read mechanism
Mechanical
Optical
Electrostatic
Thermal
According to the sensor array arrangement
Full matrix of pixels
Partial matrix of pixels
Single line of pixels
According to the data generation
Serial
Parallel

Table 1: Example classification of fingerprint sensors.

The sensor can be built as a full array. In this case, the whole fingerprint area is covered by the sensor. Alternative, the finger is passed through a scanning surface, covering at each time just a smaller surface (partial matrix) or a single line. Finally, a single pixel can be read at a time (serial), or a whole line of pixels could be read at the same time (parallel).

An example general architecture for these sensor devices is shown in Figure 3. The analog information from each sensor pixel is converted into an 8-bit value. Pixel information is provided serially at the data output of the block, although internally the information can be generated in parallel for increased speed in case the application requires it. Speed is normally limited by the time required to read a pixel.

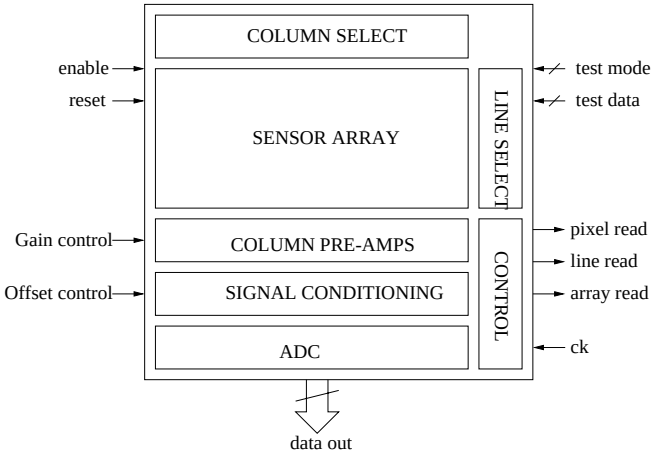

Figure 3: A general architecture for a fingerprint sensor.

The sensor generates condition signals which indicate when the data corresponding to a pixel is ready, a complete line has been read or the whole array has been read. The signal read from a pixel (or from a complete line) is amplified and the offset compensated by the column amplifiers and the signal conditioning circuitry, normally implemented by means of switched-capacitor circuitry. The sensor can also be disabled so that minimum power consumption is obtained. A reset of the sensor places the reading process at the first pixel of the array. Finally, the sensor may require a number of different test modes, feeding test data into the sensor, to check for correct functionality of the pixels or the electronic circuitry.

3.2 Data processing algorithms

A number of algorithms can be used for processing the data generated by the CMOS sensor. Some of these algorithms are different depending on the kind of application, and they can be classified as:

1. *Image formation*: these algorithms are used to form a complete image of the fingerprint under test. This is very much depending on the type of sensor. The image is directly obtained in the case of a full matrix sensor, but it needs to be reconstructed in the case of scan-based sensors.

2. *Image correction*: these algorithms attempt to correct for events such as image distortion, background noise, rotation of the finger, brightness and contrast, eventualities in the sensor (dirt), or eventual imperfections in the human fingerprint (e.g. scars).

3. *Minutiae* extraction: these algorithms identify singular points (*minutiae*) from the fingerprint image. The distance and relative positions between these points allow the generation of a fingerprint signature. This signature is then matched with the database for recognition. The actual algorithm is independent of the sensor, but can be dependent on the level of security required.

3.3 SOC general architecture

The general architecture of a SOC for these applications based on CMOS sensors is shown in Figure 4. This architecture corresponds to the bottom part of Figure 2, and it is automatically generated once the system-level specification is made available as discussed in the next section.

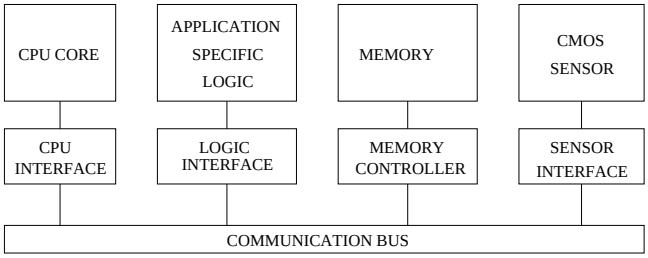

Figure 4: SOC general architecture.

The architecture is composed of CPU, Memory and CMOS-sensor cores selected according to the application requirements. An additional Application Specific Logic block can be generated depending on the application. This block is a result of the co-design approach which decides which parts are to be implemented in hardware and which ones in software as sketched in Figure 2. The block of Application Specific Logic corresponds to the parts implemented in hardware.

The blocks in Figure 4 communicate with each other via a communication bus. The interfaces between a bus and the blocks above mentioned are generated automatically.

4 System specification, validation and architecture exploration

4.1 Specification constraints

Three main parameters are considered here for determining the specification for an application based on fingerprint recognition. These are :

1. The level of security, which involves the *false acceptance rate* and the *false reject rate*. A false acceptance occurs when the system takes as known to the system a fingerprint which was not in the system database. A false reject occurs when the system does not recognize a fingerprint which should be known to the system. Both rates are related to the type of sensor mechanism used and the amount of data processing implemented.

2. The speed of the system, that is the time required to produce a go/no-go indication signal after a finger has been placed or passed through the device.

3. The cost of the system, as a SOC design, depends on variables such as the total silicon surface required, the type of sensor and thus of packaging, and the cost of the IP cores used in the implementation.

4.2 System specification and validation

The high level multi-language/multi-engine approach that we are currently using for system specification and co-simulation involves C-code and VHDL-AMS or HDL-A as shown in Figure 5. At this high level of detail, it is possible to iterate so that an adequate specification that fits the required parameters is selected. Notice that this level of detail corresponds to the top part of Figure 2, before the actual co-design and definition of the implementation take place and device cores are described in languages such as VHDL.

Each iteration includes the validation of the overall system. Validation consists of co-simulation and verification. At this level of detail, errors are found earlier in the design process and are thus easier and cheaper to correct. During co-simulation all required simulators execute concurrently. Debuggers and Graphical User Interface are launched for the control of the co-simulation and the analysis of all system parts.

4.2.1 C-code

The algorithms described in Section 3.2 are coded in C language for the system validation. Other algorithms are required for programming the system fingerprint database and for random generation of fingerprint test patterns which are used for validation of the system, and in particular, for characterizing the level of security.

4.2.2 VHDL-AMS/HDL-A

HDL-A is an Analog Hardware Description Language used for describing mixed-signal parts, the CMOS sensor, and global thermal behavior. HDL-A is an early implementation of VHDL-AMS, conceived as an extension of VHDL to cover analog and mixed-signal designs. This modeling language, used in combination with the ELDO simulator, is specially suitable for behavioral simulation of the CMOS sensor which can work in different energy domains such as the mechanical, optical, electrostatic or thermal domains. MEMS and sensor models described in HDL-A can take advantage of different levels of modeling starting at the higher level of abstraction where the model is composed of a set of equations representing relations between incoming and outcoming signals, or at a lower level of abstraction where we can use a circuit description of fundamental elements (like mechanical beams, thermal elements, electrostatic gaps and so on) to build a complete description of a multi-energy domain system. Elements are coupled at nodes representing specific energy domain signals by means of through and across variables (such as current/voltage, force/displacement, torque/rotation, heat/temperature etc ...). The final HDL-A description can be either a model of the whole sensor or a set of models for its fundamental elements. The later case is suitable for the low-level validation of the MEMS parts since it facilitates fault modeling and the injection of realistic faults [4]. This approach, in combination with fault simulation approaches for analog and mixed-signal electronic parts [8] and for digital parts using VHDL [9], allows for the evaluation of test strategies via fault simulation, the calculation of fault coverage figures and the determination of suitable test patterns. Sensor qualification can then be fully considered.

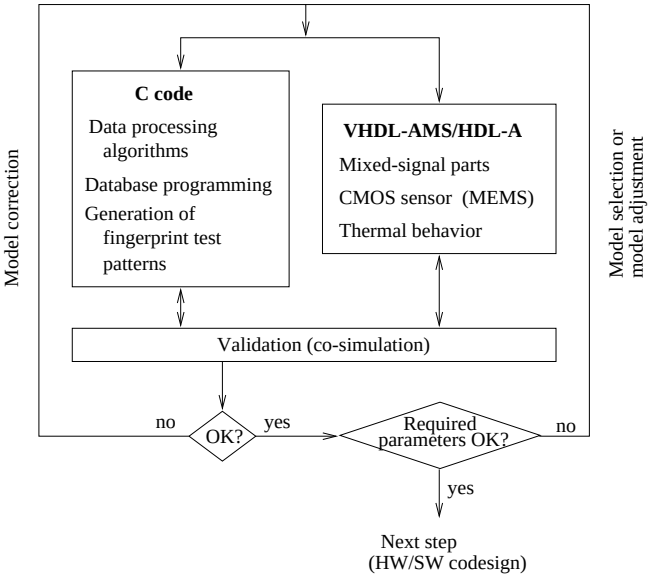

Figure 5: System-level validation and specification adjustment.

Another important multi-domain aspect that needs to be considered includes the thermal characterization of the overall device. This becomes specially important in the case of SOCs due to increasing density levels. In addition, some fingerprint sensors are either based on thermal measurements of the fingerprint or they are quite sensitive to temperature changes. Thermal characterization requires consideration of the type of packaging, and the validation of the packaging itself is very important. In fact, packaging testing is typically based on thermal characterization [10], aiming at verifying die attach quality and detecting and localizing physical defects in the heat removing path. The method is essentially based on recording the thermal response function of the structure for a step-function power excitation. A complete time-constant spectrum can be first extracted from the measured response and next the thermal resistance and capacitance map of the structure calculated [10]. Models of the thermal dynamic behavior are then built using, for example, Cauer-ladder networks. These, or higher level macromodels, can be implemented in languages such as VHDL-AMS and simulated together with the microsystem and other parts for the analysis of the thermal interactions.

4.3 Architecture exploration

The system specification must meet the constraints imposed on the cost, response time and level of security. Estimated values

for these parameters are used to guide the selection of the sensors and data processing algorithms required in the application. These parameters are not independent among each other. A faster system response will require faster sensors, CPU or memory cores which in turn will be more expensive. This selection process will be completed further down in the design flow during the hardware/software co-design when knowledge about the different cores embedded on chip is made available.

5 Implementation validation and SOC testing

Once the architecture is decided, the individual hardware and software blocs are refined by adding the necessary implementation details and constraints required for their synthesis. Co-simulation is used to ensure that the system works correctly at this implementation level, where the software is targeted to a certain processor and the communication interfaces are refined. Since more detail about the system is available, co-simulation at this stage is slower.

Although conceptually designing a SOC is analogous to the design of a printed-circuit board, access to the cores of a SOC device for testing purposes after fabrication is severely limited. Testing the device behavior is one of the most challenging tasks posed by the very high levels of integration in SOC devices. Designers must include proper mechanisms into their chips to facilitate it. Current research work on embedded-core-based testing [3] is going towards the definition of standard test mechanisms for providing access to the chip cores. This is adapted in Figure 6 for the case of the SOC architectures of our interest sketched in Figure 4.

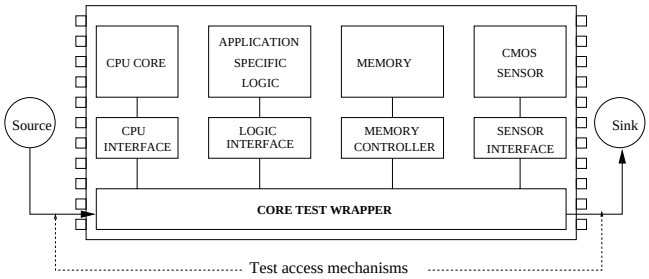

Figure 6: SOC testing.

The test approach is based around a core called *core test wrapper*, which forms the interface between an embedded core and its environment. That is, it connects the core to the rest of the SOC and to the test access mechanisms. During normal mode, the core test wrapper acts as the wanted communication bus shown in Figure 4. During testing, however, the core test wrapper can connect individually each embedded core to the test mechanisms, while keeping the core disconnected from other blocks. The test access mechanisms transport test patterns in and out of the chip, as shown in Figure 6. A test source provides the required test patterns for a core, and a test sink compares the test responses to the expected ones. Notice that test sources and test sinks could also be included in the chip for a built-in self-test approach.

The required test approach for each IP core in the system must be given along with their functionality. This is also valid for the embedded sensor, for which adequate test methods must be provided. This is indicated in the general sensor architecture of Figure 3, for which ports for applying input test data and programming different test modes are made available. Research to provide fingerprint sensors with a self-test capability is currently under way.

6 Conclusions

This paper has described our work towards the design and validation of heterogeneous systems, focusing on SOCs embedding CMOS-sensor parts, and in particular applications based on fingerprint recognition. The approach presented involves a multi-language/multi-engine design and validation scheme, where co-simulation for C-code and VHDL-AMS/HDL-A applications is taken into account at the system-level. After validation at this level is completed, a specific architecture, suiting the requirements of security, speed and cost imposed by a given application, is generated. This architecture, built from the IP cores available, can result in a mixed-technology SOC (a CMOS chip later on micromachined to complete the sensor part), or a hybrid system in which the MEMS part is produced in a separate chip. This choice depends on the type of sensor and the requirements of the application. A fully integrated approach poses significant challenges for the test and diagnosis of the device. Our work for bringing this in line with current efforts on SOC testing has also been described.

REFERENCES

[1] B. Courtois, J.M. Karam, S. Mir, M. Lubaszewski, V. Székely, M. Rencz, G. Kelly, J. Alderman, A. Morrissey, K. Hofmann, and M. Glesner. CAD, CAT and MPW for MEMS. In *Workshop on Synthesis and System Integration of Mixed Technologies SASIMI'98*, pages 207–219, Sendai, Japan, October 1998.

[2] P. Coste, F. Hessel, Ph. Le Marrec, Z. Sugar, M. Romdhani, R. Suescun, N. Zergainoh, and A. Jerraya. Multilanguage design of heterogeneous systems. In 7^{th} *International Workshop on Hardware/Software Codesign*, pages 54–58, Rome, Italy, May 1999.

[3] Y. Zorian, E.J. Marinissen, and S. Dey. Testing embedded-core-based system chips. *Computer*, 32(6):52–60, June 1999.

[4] S. Mir and B. Charlot. On the integration of design and test for chips embedding MEMS. *IEEE Design & Test of Computers*, 16(4):28–38, October-December 1999.

[5] A. Jerraya and R. Ernst. Multi-language system design. *Design, Automation and Test in Europe Conference*, pages 696–699, 1999.

[6] Thomson-CSF. *FingerChip*, July 1998. Technical data sheet FC15A140.

[7] Veridicom. *Solid-State Fingerprint Sensor*, February 1999. Technical data sheet FPS110.

[8] A.J. Perkins, M. Zwolinski, C.D. Chalk, and B.R. Wilkins. Fault modeling and simulation using VHDL-AMS. *Analog Integrated Circuits and Signal Processing*, 16:141–155, 1998.

[9] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. Karlsson. Fault injection into VHDL models: the MEFISTO tool. In *IEEE Fault Tolerant Computing Symposium*, pages 66–75, 1994.

[10] V. Székely, M. Rencz, and B. Courtois. Thermal transient testing without a tester. In *SEMICON West'98*, pages D1–D10/Section II.V., San Jose, USA, July 1998.