TECNIA, Vol.10 N°2, págs. 71-76, 2000
Universidad Nacional de Ingeniería
Lima - Perú

STUD: A SCANLET FOR TEST & UPGRADE OF DESIGNS TARGETING VERIFICATION OF FUZZY LOGIC CONTROLLERS

Luis Córdova Sosa, l.cordova@ieee.org José Paz Campaña, jpaz@uni.edu.pe
Jorge Egoávil Retamozo

Sección de Postgrado de la FIEE & Grupo de Microelectrónica de la UNI
Universidad Nacional de Ingeniería, Lima-Perú

ABSTRACT

This paper presents a preliminary work about the implementation of a Scanlet for: (a) Automatic test pattern generation targeting digital circuits based on a stuck-at-fault model, (b) in-system monitoring and debugging, (c) upgrade of a fuzzy logic controller architecture, and, (d) pin-level fault injection using the Java API (Application Programming Interface) for boundary-scan. Items (a)-(c) are verification tasks. As a first approach, the STUD (Scanlet for Test & Upgrade) system has been developed using the JTAG architecture (built-in) featured by a CPLD (Complex Programmable Logic Device) in-system configured for check and programming operations. Meanwhile, operations are executed by the Scanlet, which can be run from an Internet page. Finally, a fuzzy logic controller as an IP-core (IP stands for Intellectual Property) is used as a workbench.

Keywords: *Scanlet, ATPG, JAVA API for Boundary-Scan, Fuzzy Logic, Functional Verification.*

RESUMEN

Este artículo presenta un trabajo preliminar acerca de la implementación de un Scanlet para: (a) chequeo automático de generación de patrones relacionados con circuitos digitales basados en un modelo con interrupción en presencia de falla, (b) monitoreo y depuración in situ, (c) actualización de la arquitectura de un controlador lógico difuso, y (d) inyección de falla a nivel de pines usando Java API. Los puntos (a)-(c) son tareas de verificación. Como primera aproximación, el sistema STUD ha sido desarrollado empleando la arquitectura (incorporada) JTAG (Joint Test Action Group) y configurada por un CPLD para chequeo y operaciones de programación. Entretanto, las operaciones son ejecutadas por Scanlet, el cual puede ser corrido desde una página de Internet. Finalmente, un controlador lógico difuso, a manera de un núcleo IP, se usa como marca característica.

MOTIVATION

Since the inception of the IEEE Std. 1149.1, all major vendors and design firms found a de facto and well established protocol to help themselves improve their capabilities in terms of testability, compliance with the industry roadmap, and to extend their proprietary techniques around this test architecture. Also, Java technology is a key strategy today which has been applied to almost any known field. Java has strong disadvantages in performance but possesses more strong advantages in terms of portability, reuse, multi platform, and object oriented paradigm and high level design. The key tool –which justify completely the use of Java- is the Internet.

The main purpose of this work is to have an easy way to put vectors into the inputs pins of a FLC (Fuzzy Logic Controller) implemented in order to read the last valued control signals from the output pins of the device. This FV (Functional Verification), also can be extended to: (a) ATPG (Automatic Test Pattern Generation), (b) in-system monitoring and debugging , (c) upgrade design strategies, and (d) fault injection techniques targeting the evaluation of the system in terms of reliability. The implementation of such tasks could automate completely the fast prototyping flow of fuzzy logic hardware systems.

FUZZY LOGIC CONTROLLER

In [1] has been implemented a Fuzzy Logic Controller as an IP-core. Such task is automated and guided by the designer since the flow described there goes from a high level specification language describing Fuzzy Inference Systems (FIS for short)

to high level synthesis, customizing hierarchical VHDL-type controllers (VHDL stands for Verilog Hardware Description Language), RTL-type synthesis technology, and generating the bit stream in any format for device programming. Within this already established methodology, we can do pre and post functional and timing verification using a test bench simulating RTL vs. primitive-level description and compare high level description vs. RTL description control surfaces.

"The space of possible architectures that is explored using this methodology also needs a post-implemented verification or an in-system emulation strategy."

We have a second long term purpose targeting FLC architecture based designs which is to provide a mechanism to dynamically change or adapt the fuzzy hardware using Artificial Intelligence (AI) strategies. To accomplish these objectives we searched for some new technological concepts in the next sections.

Also, since there already exist commercial frameworks like Matlab, Simulink and Toolboxes for fuzzy logic based systems design [2,3], there is a strong need for converting a modeled system into a hardware-software fuzzy system and for verifying the implementation as well.

Fig.1 shows the control surface of an implemented Mandani-type FLC. This surface can be generated using the hardware description and running the test bench in a commercial HDL (Hardware Description Language) simulator [9].

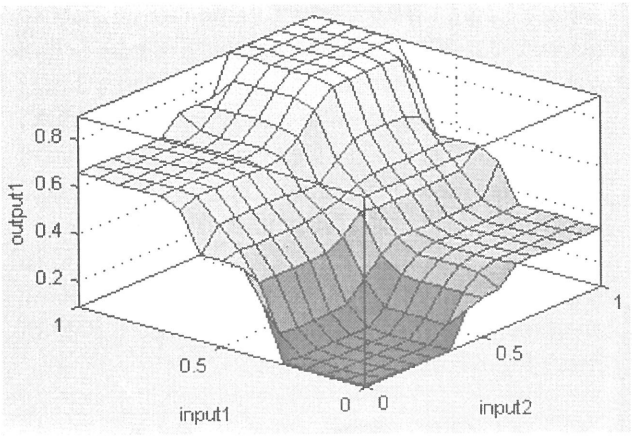

Fig. 1: Control Surface characteristic of the FLC.

INTERNET-DRIVEN PLD SYSTEMS

The concept of internet-driven PLD (Programmable Logic Devices) systems is becoming more important in today networked-based applications. The ability for such a system to reconfigure itself by connecting to a server and interacting through the network with other sub-systems is a key process to turn them from static hardware into adaptive systems.
Using the well known IEEE Std. 1149.1 Architecture, a.k.a. JTAG standard is an efficient
way to interact with the hardware part of our implemented system. Some benefits are: The reduced pin count used, the embedded strategy for test and sample executions, simple serial protocol, low power architecture, and widely supported.

The main idea of a networked I-System (Internet System) is shown in Fig.2. The task of connecting both the Internet and the hardware prototype is fulfilled by the I-System. The latter is itself a Java application running on a JVM (Java Virtual Machine). The JVM is supported by an operating system running in any hardware system, namely PC, Workstation, ATE (Automated Test Equipment) station, etc.

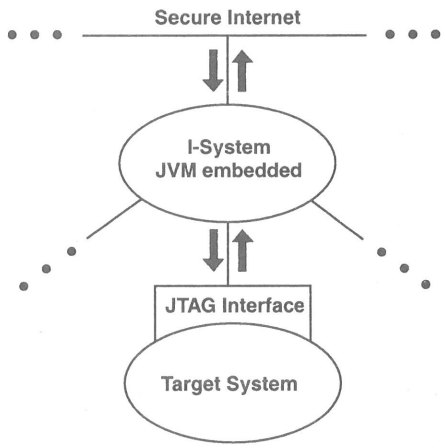

Fig. 2: General Scheme of a Networked I-System (Internet System).

JAVA API FOR BOUNDARY SCAN

In order to enable a reuse Methodology developing Java based applications there exist the so called Java API's. Likewise, there is the Java API for Boundary-Scan which enables developers to create a Scanlet: a Java-based scan application. The API for boundary scan implements the standard micro-scan operations and defines the data types for the hardware interface using the JTAG [5] Architecture. Since all classes are written in Java language, they can be run in any commercial browser regardless of the operating system or platform.

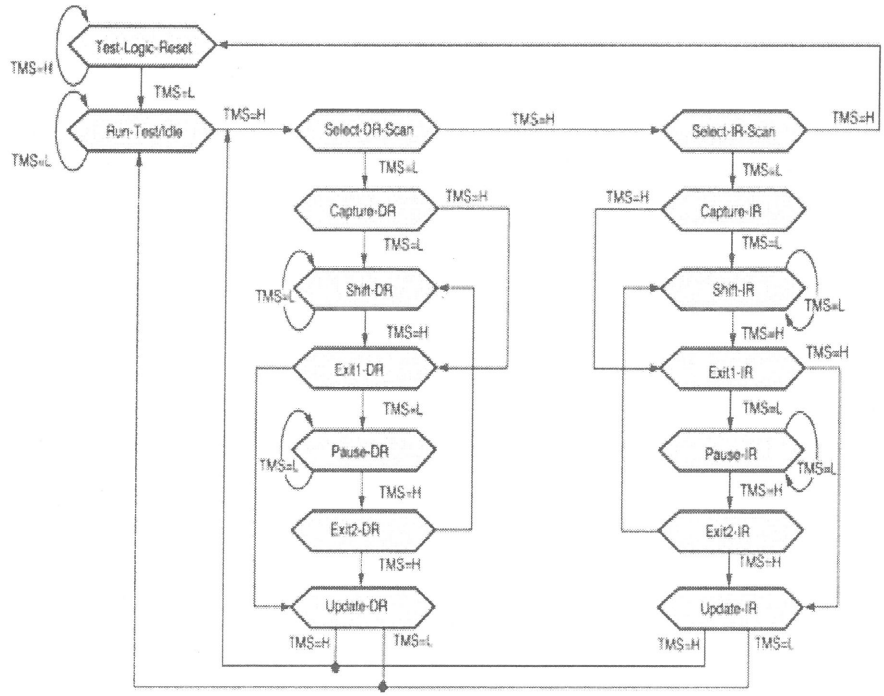

Fig. 3: TAP (Test Acces Port) finite state machine diagram.

Fig.3 shows the TAP (Test Access Port) controller state machine of the JTAG protocol. Writing a Scanlet code means integrate micro-scan operations in order to execute some instruction in a desired order using specific data.

Object Paradigm

Java technology also enable us to handle application at a high level of abstraction, e.g., complex systems. According to Fig. 4, we can implement our hardware system employing the object oriented paradigm, having in mind the dealing with ever growing complexity of nowadays designs and the reuse of IP challenges.

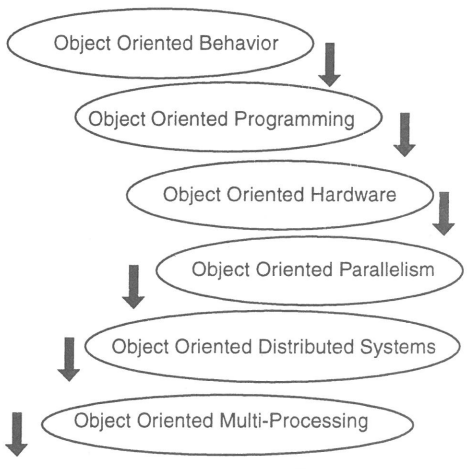

Fig. 4: Scheme of the O^2 Paradigm.

SCANLET

The architecture of a Java scan application shown in Fig.6 consist in 5 levels of functionality. Such levels are (from top to bottom): A Scanlet, the Java API for boundary-scan, a specific dynamic link library, a hardware driver (parallel port), and the Device Under Test (DUT). The SPTM (Scan Path Testing Methodology) is based upon the use of such components in order to fulfill specific tasks.

These tasks could be: apply and read test vectors, debugging and monitoring, fault injection process, etc.

STUD: A Scanlet for Test & Upgrade of Designs

Fig.5 shows the set up of the plant for experimentation tasks described in Section 5: a host PC (WindowsNT O/S, Internet Explorer, JRE (Java Runtime Environment – JVM embedded) connected to the Internet using a parallel port (a parallel cable with embedded circuitry for the signal adaptation), and a target board with a JTAG port and target device (a CPLD). The latter is JTAG compliant.

Fig. 7 shows the Scanlet scheme. The upper left side illustrates the already automated design flow of the FLV. The latter flow gives us three

specification formats: (a) EDIF (Electronic Design Interchange Format) description of the design, (b) the bit stream, namely the configuration data of the design, and (c) the test bench automatically generated within the design flow explained earlier. Taking the data from the test bench output, it is straightforward to plot the surface control using a function written in Matlab.

ATPG & EMULATION

With the EDIF we can develop a Matlab program that can handle such an input file that can perform the analysis and generation of test vectors for a particular circuit description. The implemented algorithm is based on a stuck-at fault model that computes simultaneously values for fault coverage and processing performance. Also, it is useful to have emulation through merge data read from hardware with data stimuli from Matlab.

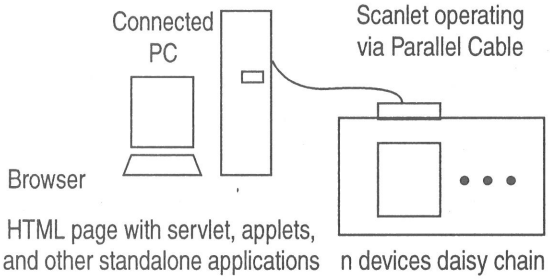

Fig. 5: Set up of the plant.

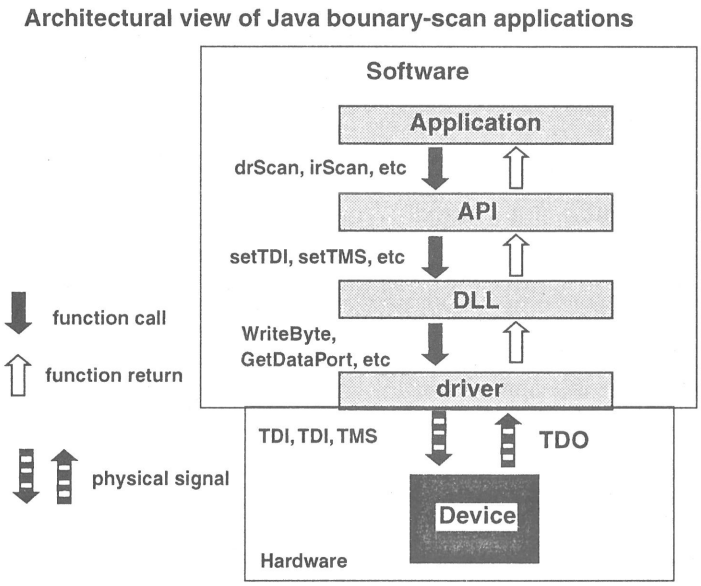

Fig. 6: Architecture of a Scanlet.

EXPERIMENTATION

Three experiments were performed. Such experiments comprise the execution of different applications (programs in JAVA). The experimental plant is described as in Fig. 5 and the experimentation flow is as stated in Fig. 7, which shows a header above indicating a browser pointed at a defined URL (an Internet address). Fig. 7 also shows a device wich is actually a board embedded with a XC95108 device. The following operations were successfully accomplished:

1st Experiment:
Programming the CPLD XC9500 family through JTAG with the design configuration using the DOS command line prompt, calling Java classes with required arguments.

2nd Experiment:
Running stand-alone operations targeting the XC9500 using different Scanlets for each operation. (reading, sampling signals, composition of two or more simple operations and starting to manage daisy chain operations).

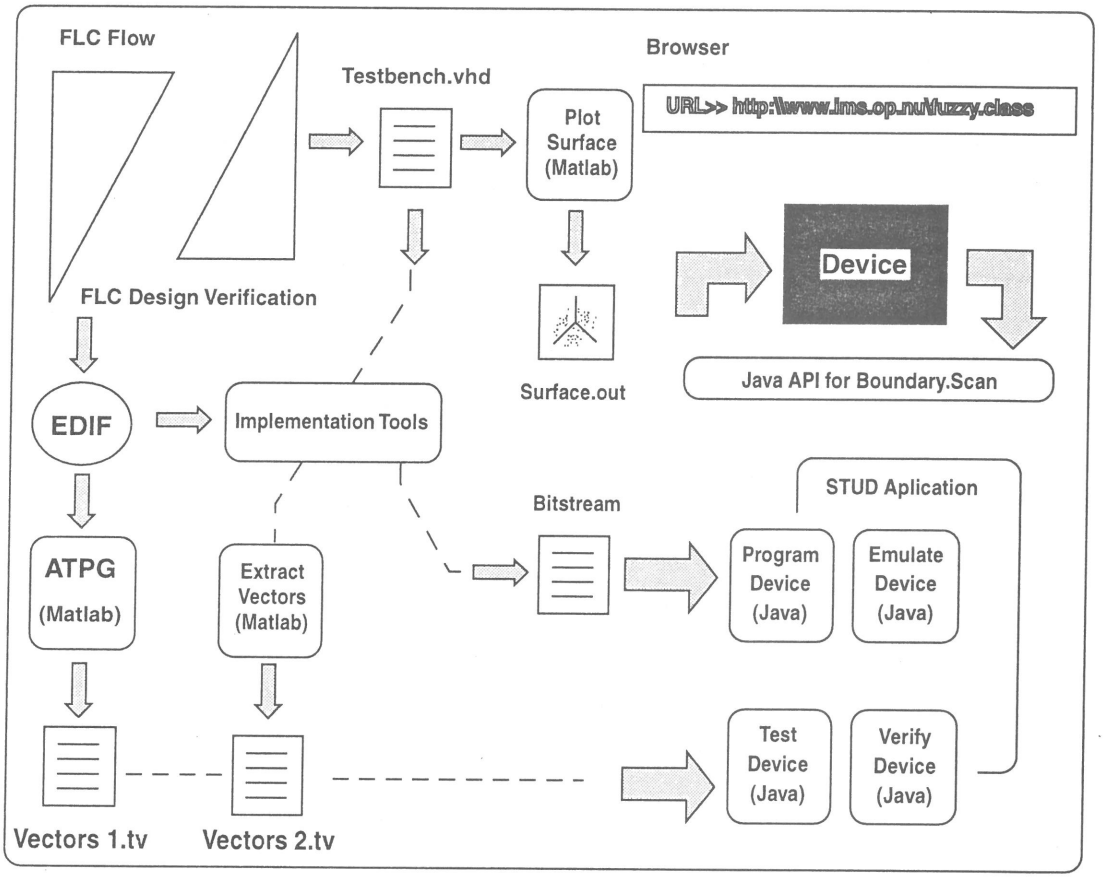

Fig. 7: Scanlet Scheme.

3rd Experiment:
Run the Scanlet in the built-in JVM (Java Virtual Machine) implemented in a commercial browser (Internet Explorer). Also, run the Scanlet from the laboratory as a stand alone application using a JVM for the PC.

Material Description:
Development board from XESS XS95 v1.2 + socket PLCC 84 + Parallel Download Cable III + Oscilloscope + Multitester + etc.

EDA Tools:
JDeveloper, WebPack v2.1i. IMPORTANT: this tool generates files in format *.xbt when generating other files like *.jed. The *.xbt is used to generate the *.dat file employing the xbt2dat compiled class from Xilinx [7]. It is also helpful to have fully operative the WebPack JTAGProgrammer module, which can be used for operations verification and for other useful tasks.

Device:
XC95108 PC84 -7 embedded on the XS95 development board from XESS. JTAG Architecture, which is built-in. (IEEE Std.1149.1X compliant).

JTAG built architecture description:
BSDL (Boundary Scan Description Language, which is a subset of the VHDL IEEE Standard.) file (xc10884.bsd)

Java files:
xbt2dat.java , xc9500.java , JAVA API for boundary-scan and other classes from Xilinx (some compiled but with no available code).

First workbench design:
FLC.* (*.vhd, *.ucf, *.xbt, *.jed, *.svf, *.dat).

Basic Operations:
Read levels, Sample levels, Compare levels, manage of files and URLs (in the *.pack, *.dat compressed or decompressed format respectively).

Device data versions and other characteristics:
ver1 = ver2 (*.bsd)
device version = ver1
access_pins_84 => controlr_bc_jtag (bc = boundary-scan cells) using the xc10884.bsd (more specific).

Useful Information
- Data sheet XC95108 PC84 (pdf at Weblinx).
- XS95 v1.2 manual (pdf at Xess [8] website).

CODE & CURRENT WORK

The Matlab code targets the ATPG while the Java code is the application itself representing the Scanlet. The following are some parts of the code and the list of implemented Matlab functions.

Java code (Scanlet):

Method terminate:

```
public static void terminate() {
        javaScanObj.irScan( ispex );
        javaScanObj.irScan( bypass );
        System.out.print(«Completed...\n» );
}
```

Method getIDCODE:

```
public static xilinxCpldBits getIDCODE( ) {
      idcode = new xilinxCpldBits( (byte) 0xfe );
      deviceIDInput = new xilinxCpldBits( (int)
      0xffffffff );
      xilinxCpldBits deviceID=new
      xilinxCpldBits((int) 0xffffffff );
      // read the IDCODE
      javaScanObj.irScan( idcode );
      javaScanObj.drScan( deviceIDInput, deviceID );
      return( deviceID );
}
```

Matlab code (ATPG functions):

- Atpg_main.m
- Atpg_parser.m
- Atpg_reader.m
- Atpg_generate.m
- Atpg_observability.m
- Atpg_controlability.m
- Atpg_propagation.m
- Atpg_retropropagation.m

Current work retarget such functions in order to implement fault injection techniques.

CONCLUSIONS

We have designed an automated flow for the development of patterns and medium programs for debug, monitoring and test of attached hardware systems. There is ongoing work in ATPG functions in order to implement fault injection techniques and porting to JAVA all the algorithms. It needs to be further researched how to change configuration parameters for implementation of adaptive neuro-fuzzy controllers. On the other hand, we are targeting projects within the area of Internet appliances too.

ACKNOWLEDGEMENTS

This work was supported by the Department for Graduate Studies of the Faculty of Electrical and Electronic Engineering of the UNI. We would like to thank the Microelectronic Group of the UNI and give special thanks to Neil Jacobson, Patrick Kane and Anna Acevedo from Xilinx, Inc.

REFERENCES

1. Córdova Luis, Egoávil Jorge. Controlador Difuso Realizado en un FPGA: Aplicación. VI Iberchip Workshop, Pages. 300-305, Sao Paulo-Brasil, 2000.
2. Nuñez C.; Rojas A., Córdova L. Diseño e Implementación de un Controlador Difuso TSK Utilizando Sintonía ANFIS. Aplicación: Péndulo Invertido. III Jornada Iberoamericana de Inteligencia Artificial en Sistemas Eléctricos JIASE, Vol.N°03, Pages.734-740, Lima-Peru, 1999.
3. Córdova L., Egoávil, J., Salazar R., Escudero M. FuzzyChip. Intercon'99 – Memoria de Concurso de Proyectos, Lima-Peru, 1999.
4. Córdova L., "Design of Fault-Tolerant Circuits Hardening Memory Elements in VHDL", Design Automation Conference 2000, June 5~9, Los Angeles, CA, USA.
5. EETIMES, Signals, "Cell Phone Challenges Designers" , June 5, 2000.
6. Córdova L. Intelligent Microelectronics Systems Group. *http://207.17.220.225/ims/index.html , 2000.*
7. Xilinx, Inc. *http://support.xilinx.com/apps/fpga.htm*
8. Xess Corp. *http://www.xess.com/prod002.html*
9. ModelSim, Inc. www.model.com & Innoveda www.innoveda.com