Kasetsart J. (Nat. Sci.) 36 : 212 - 218 (2002)

JAVA API for Boundary Scan of FPGA System

Mongkol Asawaplungkul[1] and Pradondet Nilagupta[2]

ABSTRACT

Nowadays boundary-scan technique is widely supported by most semiconductor vendors and test system companies. A boundary-scan test bus, sometime called boundary scan is the standard test access port, can now enabled in the FPGA System by adding a few line of VHDL code. This paper presents the designed PC-interface circuit and the designed boundary scan API, which implemented in JAVA to provide an easy way for accessing the test access port of FPGA device. Using the designed API equipped with some low-cost PC hardware interface help us to conquer the complexity task of testing digital system that was designed based on the FPGA devices and it is an alternative method for reducing the test cost of system.

The four basic classes and interfaces JavaScanOperations class, JavaScanState class, JavaScanBitIf class, and JavaScanHWIf class are defined as well as an extended JAVA API interface called ScannedObject has been constructed. We test the software on a XC4010E development board. The experiment shows that the result from the boundary-scan port is correspondent to the real-life of the circuit. The maximum scanning frequency equals to 3.3574 KHz which depends on the FPGA device that used in the circuit.

Key words: JAVA, FPGA, IEEE 1149.1, JTAG, design for testability, boundary scan

INTRODUCTION

The appearance of smaller packages for VLSI components has created new challenges in testing electronic circuit assemblies. Since an increasing number of layers in the PCB, new SMT techniques such as Tape Automated Bonding (TAB), and Ball Grid Arrays (BGAs) packages, pressure for fast product launch and quick delivery. Test engineers find themselves under a continuous pressure to adopt effective test strategies, develop applications quickly, and be expedient in sending good products out. One disadvantage in testing small devices is that the test procedure gets exponentially more complex. When circuit boards were large, we tested them with techniques such as bed-of-nails. The problems with bed-of-nails testing were exacerbated as board dimensions got smaller and surface-mount packaging technology improved. If devices were mounted on both sides of a circuit board, no attachment points were left for the test equipment. To solve these problems, a group of European electronics companies formed a consortium in 1985 called the Joint Test Action Group (JTAG). The consortium devised a specification for performing boundary-scan hardware testing at the IC level. Today, boundary-scan technology is probably the most popular and widely used design-for-test technique in the industry. (Oshana, 2002)

The Boundary-Scan standard, formally known as IEEE/ANSI Std 1149.1-1990 which inspired from the Scan Path Technique (JTAG technology B.V, 1997) sometime called JTAG.

[1] Information Technology Department, Seagate Technology (Thailand) Ltd., Samutprakarn 10270, Thailand.

[2] Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Bangkok 10900, Thailand.

Boundary-Scan consists in insuring observability and controllability of a board through its edge connector using non-contact methods. The boundary-scan based access to PLD device can be used for testing or in-system programming operations. This paper will focus on testing application based on a standard interface IEEE 1149.1 Design-for-Test (DFT) technique which transforms the in-circuit test methods of the hardware PCB that consist of many ICs into a software based techniques. A set of test patterns are then presented through dedicated test inputs allowing full observability and controllability of every pin of every ICs on the board. This methodology of incorporating DFT allows complete control and access to the boundary pins of a device without the need of expensive test equipment. Each IEEE 1149.1 compliant device includes a boundary-scan cell on each input, output or bi-directional device pin that under normal condition is transparent and inactive allowing signals to pass normally. When the device is placed in test mode, input signals can be captured for later analysis and output signals can be set to affect other devices on the board.

Today a new technology that extends the functionality of smallest Java virtual machine provide a blueprint to help developers rapidly generate applications for the dynamic testing, debugging, and reconfiguration of programmable logic devices (PLDs). The Java API can be deployed by any PLD supplier, as well as manufacturers of automatic test equipment, device programmers and software developers. The new API for boundary scan will bring a new level of functionality and compatibility to chip set and electronic board design. It provides a solution for unifying the generation and distribution of programs required for configuring, testing, and debugging in system programmable devices and boundary-scan chips that are compliant with the IEEE 1149.1 standard. The Java(TM) technology enabled API for Boundary-Scan is a licensee open class that describes a programming interface appropriate for developing applications that communicate with systems that support IEEE Std 1149.1. This Java(TM) Code enabled API facilitates fully portable, network friendly system development to allow for both local and remote field upgrade test and debug of complex electrical systems.

METRIALS AND METHODS

Boundary scan architecture

A boundary Scan architecture shown in Figure1 is divided in two parts: a control part called “TAP controller” and a data path composed of several registers accessible serially through TDI pin and TDO pin for controllable and observable properties (LALA, 1997). The necessary and sufficient test pins has been added, there are 4 mandatory test pins, plus one optional. These pins are called “Test Access Port”, TAP, compose of: TDI (Test Data Input) is a serial input pin. TDO (Test Data Output) is a serial output pin. TCK (Test CLOCK) is a clock signal that synchronizes the 1149.1 internal state machine operations. TMS (Test Mode Select) is the internal state machine mode select signal. TRST (Test RESET) is an optional reset pin that operates asynchronously to reset the TAP Controller.

The control part in boundary scan is called TAP Controller (Test Access Port Controller). TAP Controller is a finite state machine with two input signals TMS and TCK as shown in Figure 2. TAP Controller state diagram have two columns with similar labels for each position the suffix, IR (Instruction Register), and DR (Data Register). Each position represents a TAP controller state. The diagram shows how test data is received and applied to and from the IEEE1149.1 circuitry. This diagram applies to all IEEE1149.1 compliant devices (Maunder, 1990). We can simplify the state of the TAP state diagram into five basic operations Test-Logic-Reset, Run-Test/Idle, Capture, Shift, and Update.

Boundary-Scan data path composed of

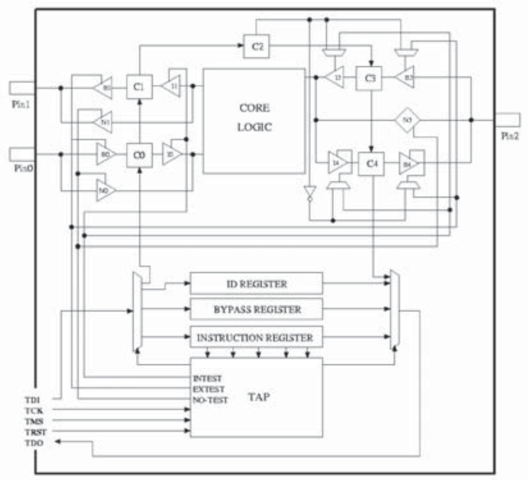


Figure 1 Boundary-scan architecture.

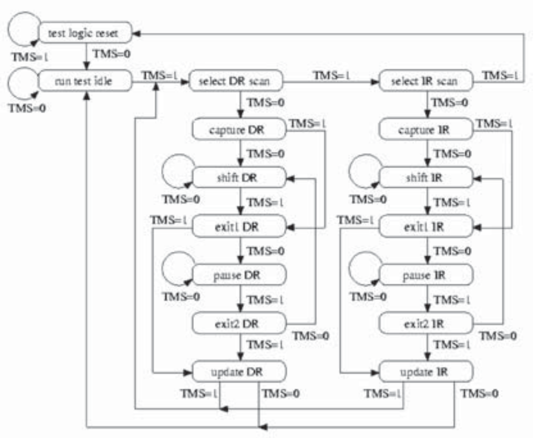


Figure 2 TAP Controller state diagrams.

several registers which contain three register that are mandatory: IR (Instruction Register), Bypass Register and the boundary scan register. Other register are optional like the identification register (ID) and user-specific register (USER).

The boundary scan instructions consist of the following instructions: BYPASS instruction holds the chip logic out of the scan chain. SAMPLE/PRELOAD instruction is for takes a “snap-shot” of the I/O logic-levels, the latter allows preconditioning output cells. EXTEST is the instruction that tests for external inter-connection short/opens. It impedes normal device functionality. The EXTEST, BYPASS, SAMPLE /PRELOAD are mandatory, while other instructions are optional i.e. INTEST is an instruction for testing the core logic function of the circuit. IDCODE is an instruction for shifting out the content of the identification register, locating the chip on the board and recognize it. RUNBIST is an instruction for stimulating and reading self-test registers. USERCODE is an instruction use for identifying PLDs mainly. CLAMP is an instruction use for avoiding bus contention by presetting output values, sets Bypass register between TDI and TDO. HIGHZ is an instruction similar to CLAMP but leaves outputs in high-impedance.

The XC4000E family is the FPGA devices (Xilinx Press, 1998) that we used in this work supported all mandatory boundary scan instructions specified in the IEEE 1149.1. The public boundary scan instructions are always available prior to configuration but after configuration, the instructions and any USER instructions are only available if the boundary scan specified in the schematic/HDL code. During configuration, the SAMPLE and BYPASS are available but it is recommended that not to perform boundary scan operations during configuration period. If Boundary Scan is not used after the device is configured, the user can use the special boundary pads as input or output pins.

Boundary-Scan operation is independent of individual IOB (I/O Block) configuration and package type. The technique to provide the maximum flexibility for interconnect testing is treated all IOBs as independently controlled bi-directional test capability even after configuration, including any unbonded IOBs. It should be noted that during configuration of the FPGA device by using standard programming pins be sure not to toggle the TAP pins, since inadvertent toggling of the TAP pins can turn on the boundary scan circuitry. The FPGA devices from Xilinx have some deviation of the boundary scan circuit from the IEEE standard in that the CCLK, PROGRAM\, and DONE pin are not scanned and the operation of the TAP controller cannot be affected by boundary-scan test data. Additionally, the EXTEST instruction with the INTEST-like functionality is not specified in the standard, and system clock inputs are not disabled during EXTEST, as recommend.

Hardware and software

The XC4010E Xilinx's FPGA is used for implementing the circuit as well as the Xilinx Foundation Series version 1.5 is used for design and implement the circuit into Xilinx FPGA. VHDL is the language that used to design the digital circuit. The Xilinx Demonstration board and XChecker cable is the hardware tools for download design from Xilinx Foundation Series into FPGA chip and observe the circuit result.

The interface part between a computer and a boundary scan port of Xilinx FPGA device is a parallel port and some additional low-cost hardware. Figure 3 shows a schematic diagram of a circuit between parallel port and the boundary scan port of the FPGA device.

The application programming interface (API) is built in Java language because the real world of semiconductor device and its operations are most logically presented in terms of objects and associated methods to follow the Java object oriented paradigm. The advantage of object-oriented paradigm is easy to extend and manage complexity of the system.

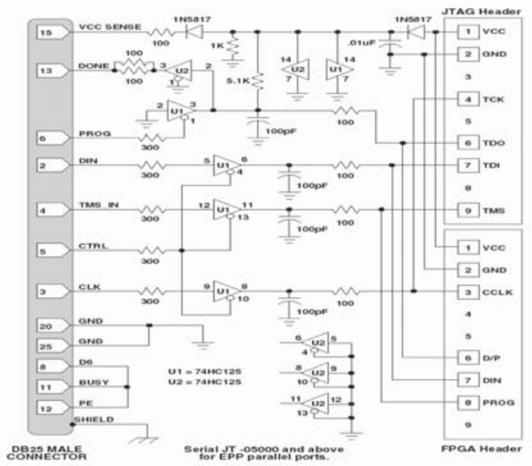


Figure 3 A Schematic diagram of the interface circuit.

FPGA Boundary Scan enable framework

Even the Xilinx's FPGA device already provide the external pins for boundary-scan standard, by default the internal boundary-scan circuit is disabled. While using Xilinx foundation software to design the FPGA-based system, we can enable the boundary-scan circuit of FPGA device by adding the following two part of code to the top level of VHDL system code, as shown in Figure 4.

```
--** insert the following after the begin keyword**
U1: BSCAN port map (TDO => TDO_NET,
                    TDI => TDI_NET,
                    TMS => TMS_NET,
                    TCK => TCK_NET);
U2: TDI port map (I =>TDI_NET);
U3: TCK port map (I =>TCK_NET);
U4: TMS port map (I =>TMS_NET);
U5: TDO port map (O =>TDO_NET);
```

```
--** insert the following between the architecture
-- and begin keywords**

signal TCK_NET  : STD_LOGIC;
signal TDI_NET  : STD_LOGIC;
signal TMS_NET  : STD_LOGIC;
signal TDO_NET  : STD_LOGIC;


   component BSCAN
     port (TDI, TMS, TCK out STD_LOGIC;
           TDO: in STD_LOGIC);
   end component;
   component TDI
     port (I: out STD_LOGIC);
   end component;
   component TMS
     port (I: out STD_LOGIC);
   end component;
   component TCK
     port (I: out STD_LOGIC);
   end component;
   component TDO
     port (O: out STD_LOGIC);
   end component;
```

Figure 4 Boundary scan enabler.

Basic Boundary Scan Java API Class diagram

A class diagram is act like a blue-print of the software system. The class diagram of the basic Java API shown in Figure 5 shows all classes in the package, relationship and dependency between classes.

There are four basic classes and interfaces that make up the Java API for boundary scan contains JavaScanOperations class, JavaScanState class, JavaScanBitIf class, and JavaScanHWIf class. The instance of class JavaScanOperations is the object describes all basic boundary-scan operations, while the javaScanBitIf is the interface describes the mechanism for access the boundary-scan test or programming data. Given the operation of the 1149.1 state machines the number of operations supported by the API are actually quite simple and can be simply enumerated which is the responsible of the JavaScanState class. The instance of JavaScanState is object which describes the 16 states of the TAP controller state machine. The JavaScanHWIf is interface which describes the mechanism for generating the actual electrical signals to stimulate the TAP through the application port, such as parallel port discussed. The Java manner to interface to a wide variety of disparate devices is implemented as a Java Native Interface (Java Student Guide, 1999). The dynamic linked library that is provide from the third party interface cable vendors allow us to wrap it as a Java object by using JNI, resulted in JavaScanHWIf class, this allow us to change the interface type (such as USB, Firewire).

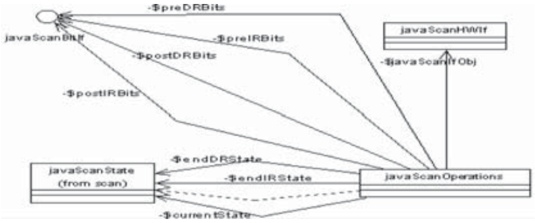
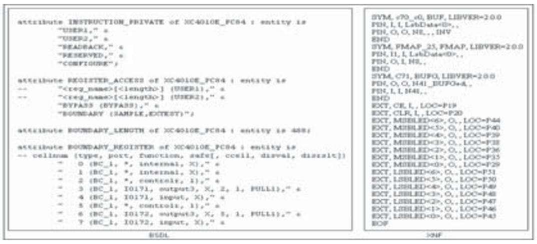


Figure 5 Class diagram of the basic Java API.

Extended Java API for boundary scan interface

The basic API provide the capable to interface with the boundary-scan port of the FPGA devices but the change in FPGA chip number or chip functionality result in the need to change the code. The common need input into the Java code of the test system that uses the basic API is the mapping information of boundary-scan cell number and physical pins number of FPGA device. Other information is the mapping between signal name and physical pins of the FPGA device. The first information can be get from an external file known as the BSDL (Boundary-Scan Description Language) file which defines the capabilities of any single device's boundary-scan logic. These files are supplied by the IC manufacturer. The second information can be get from the external file that generated by Xilinx Foundation Series Software known as XNF (Xilinx Net list File) file which defines the mapping between signal name and the physical pin number of FPGA device. Every FPGA device must has its individual file. An example of the portion of BSDL file and XNF file are shown in Figure 6.

The extension of API concept results in one additional object called ScannedObject. The ScannedObject has responsibility for initial its internal hash table with mapping information between pin number and boundary scan cell number by travel along the BSDL file and grasp the information, also initial its internal hash table with mapping information between signal name and



```
attribute INSTRUCTION_PRIVATE of XC4010E_PC84 : entity is
        "USER1," &
        "USER2," &
        "READBACK," &
        "RESERVED," &
        "CONFIGURE";

attribute REGISTER_ACCESS of XC4010E_PC84 : entity is
--      "<reg_name>[<length>] (USER1)," &
--      "<reg_name>[<length>] (USER2)," &
        "BYPASS (BYPASS)," &
        "BOUNDARY (SAMPLE,EXTEST)";

attribute BOUNDARY_LENGTH of XC4010E_PC84 : entity is 488;

attribute BOUNDARY_REGISTER of XC4010E_PC84 : entity is
-- cellnum (type, port, function, safe[, ccell, disval, disrslt])
        "    0 (BC_1, *, internal, X)," &
        "    1 (BC_1, *, internal, X)," &
        "    2 (BC_1, *, controlr, 1)," &
        "    3 (BC_1, IO171, output3, X, 2, 1, PULL1)," &
        "    4 (BC_1, IO171, input, X)," &
        "    5 (BC_1, *, controlr, 1)," &
        "    6 (BC_1, IO172, output3, X, 5, 1, PULL1)," &
        "    7 (BC_1, IO172, input, X)," &
```

BSDL

```
SYM, s70_s0, BUF, LIBVER=2.0.0
PIN, I, I, LabData<0>, ,
PIN, O, O, N8, , , INV
END
SYM, FMAP_25, FMAP, LIBVER=2.0.0
PIN, I1, I, LabData<0>, ,
PIN, O, I, N8, ,
END
SYM, C71, BUFG, LIBVER=2.0.0
PIN, O, O, N41_BUFGed, ,
PIN, I, I, N41, ,
END
EXT, CK, I, , LOC=P19
EXT, CLR, I, , LOC=P20
EXT, MSBLED<6>, O, , LOC=P44
EXT, MSBLED<5>, O, , LOC=P40
EXT, MSBLED<4>, O, , LOC=P39
EXT, MSBLED<3>, O, , LOC=P38
EXT, MSBLED<2>, O, , LOC=P36
EXT, MSBLED<1>, O, , LOC=P35
EXT, MSBLED<0>, O, , LOC=P29
EXT, LSBLED<6>, O, , LOC=P31
EXT, LSBLED<5>, O, , LOC=P30
EXT, LSBLED<4>, O, , LOC=P49
EXT, LSBLED<3>, O, , LOC=P48
EXT, LSBLED<2>, O, , LOC=P47
EXT, LSBLED<1>, O, , LOC=P46
EXT, LSBLED<0>, O, , LOC=P45
EOF
```

XNF

Figure 6 An Example of the portion of BSDL and XNF file.

physical pin by travel along the XNF file. The class diagram of ScannedObject with other class shown in Figure 7.

RESULTS AND DISSCUSSION

The complete class diagrams of basic Java boundary scan API is shown in Figure 8. It provides the capable to interface with the boundary-scan port of the FPGA devices.

The JavaScanOperations class contains the basic methods to describe the required boundary-scan program, test and debug operations. The JavaScanState class describes the 16 states of the IEEE Std. 1149.1 Test Access Port Controller. The javaScanBitIf class describes the interface methods for accessing application specific data. The implementation of this interface's methods comprehends the desired data compression and decompression algorithms. The JavaScanHWIf class describes the interface to boundary scan hardware.

The extended API provides the transparent mapping between signal name of the VHDL circuit design and internal boundary-scan cell number. The complete class diagram includes an implementation of extended API is shown in Figure 9 which shows all available operations.

The functional test process for this API is done by generate several simple sequential circuit such as flip-flop, counter in VHDL by using Xilinx Foundation version 1.5 and download to Xilinx FPGA demonstration board. To read the result from the boundary-scan port of FPGA device, we write a simple Java program that use this API and use the sample instruction of the boundary-scan port. We found that the result from the boundary-scan port is correspondent to the real-life of the circuit. The only limitation is the speed of parallel port since it works in SPP mode. The maximum speed of SPP mode for receiving data is 200 Kbytes per second and device that used in the board is XC4010E (PC.84) has boundary scan cell 488 cells. The

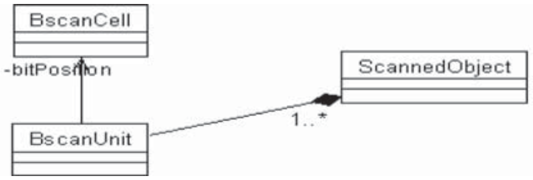


Figure 7 Class diagram of ScannedObject.

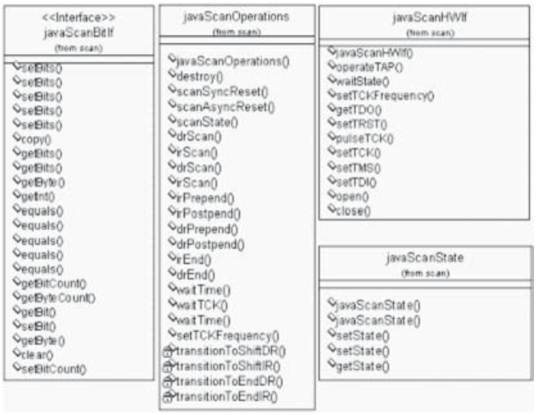


Figure 8 Class diagrams of basic API.

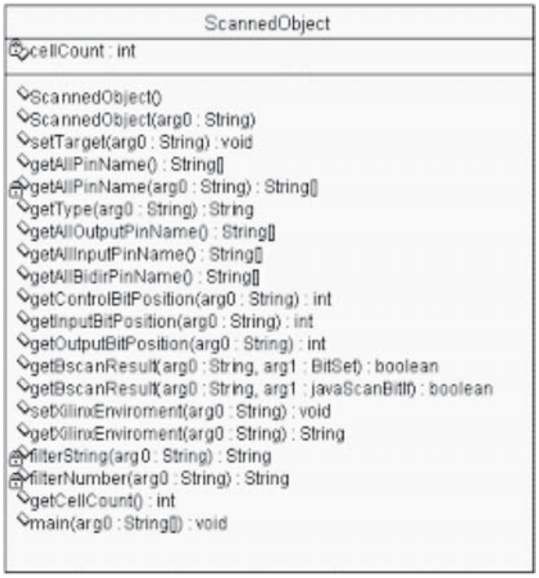

ScannedObject
cellCount : int
ScannedObject()
ScannedObject(arg0 : String)
setTarget(arg0 : String) : void
getAllPinName() : String[]
getAllPinName(arg0 : String) : String[]
getType(arg0 : String) : String
getAllOutputPinName() : String[]
getAllInputPinName() : String[]
getAllBidirPinName() : String[]
getControlBitPosition(arg0 : String) : int
getInputBitPosition(arg0 : String) : int
getOutputBitPosition(arg0 : String) : int
getBscanResult(arg0 : String, arg1 : BitSet) : boolean
getBscanResult(arg0 : String, arg1 : javaScanBitIf) : boolean
setXilinxEnviroment(arg0 : String) : void
getXilinxEnviroment(arg0 : String) : String
filterString(arg0 : String) : String
filterNumber(arg0 : String) : String
getCellCount() : int
main(arg0 : String[]) : void

Figure 9 Class diagram of extended API.

number of boundary-scan cell depends on the FPGA device that used in the circuit. In this case, we can calculate the maximum scanning frequency equals to

$$\frac{200\,\frac{\text{Kbytes}}{\text{sec}} \times 1024\,\frac{\text{Kbytes}}{\text{sec}} \times 8\,\frac{\text{Kbytes}}{\text{sec}}}{488\text{ bits}} = 3.3574\text{kHz}$$

This speed limitation can eliminate by using the faster interface cable which based on another standard interface port such as USB or Firewire. We still can use this API after change interface type by replace the old dynamic linked library with the new one since the JavaScanHWIf class use the native interface implementation.

We can use this API package for create a kind of testing application that automate and aided the FPGA testing process.

CONCLUSION

In this paper we present the basic Java API and extended version for boundary-scan of Xilinx FPGA. The Xilinx FPGA system that applies the boundary-scan must be Designed-For-Test. We also present the way to make the Xilinx's FPGA-based design to be a Designed-For-Test Chip. The future work will use all these API for implement Test aided equipment for FPGA-based system.

LITERATURE CITED

Java Student Guide. 1999. Java Native Interface. Java Programming Language. Sun Micro-Systems Inc, Broomfield, Colorado.

JTAG technology B.V. 1997. Boundary-scan Test & Programming solution. JTAG technology B.V., San Jose, CA. 14p.

Lala. P.K. 1997. Design of Testable Sequential Circuit: Digital Circuit Testing and Testability. Academic Press, New York. 199p.

Matos, S. C. Leao, and C. Ferreira. 1993. Control and observation of analog nodes in mixed-signal boards, pp.323-330. *In* Proceedings of International Test Conference.

Maunder C.M. and R.E. Tulloss. 1990. The Test Access Port and Boundary Scan Architecture. IEEE Computer Society Press, Los Alamitos, California. 398p.

Skahill K.1996. VHDL for Programmable Logic, Addison Wesley. 593 p.

Xilinx Press. 1998. XC 4000E Products. The Programmable Logic Data Book. Xilinx Corp., San Jose, California.

Received date : 26/11/01
Accepted date : 05/05/02