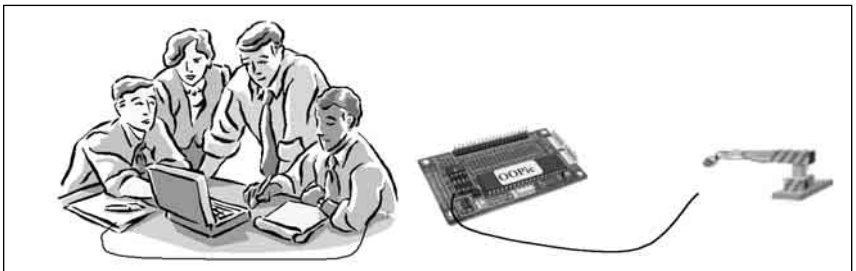
Gary F. Templeton

Object-Oriented Programming of Integrated Circuits

New programming architecture is revolutionizing how programmers control hardware circuitry.

The promise of field programmable integrated circuits (FPIC) is that they may someday allow end users to tailor hardware devices to their needs. But developing FPICs has traditionally been arduous, and procedural programming interfaces have limited the expansion of the installed base of ICs [2]. Enter Scott Savage, founder and president of Savage Innovations (www.oopic.com), who has developed the first object-oriented methodology for encoding FPICs. The new programming architecture, named OOPIC (Object-Oriented Programmable Integrated Circuits), is revolutionizing how programmers and the applications they develop control hardware circuitry. The general-purpose OOPIC device facilitates hardware programming using common OO programming languages (Visual Basic, Java, or C). The technique, which integrates modern approaches from industrial, computer, electrical, and mechanical engineering, is already being taught in a handful of universities internationally, such as Camosun University in Canada, and Purdue and Princeton Universities in the U.S. OOPIC has spawned at least one company, Magnevation (www.magnevation.com/), as well as collaborations between Savage Innovations and several others.



The structure of the OOPIC development environment.

The integrated circuit, one of the most technologically influential innovations of the twentieth century [3], is used primarily to develop analog and digital logic gates and microprocessors. These circuits have greatly expanded the field of computer science and the functionality of a wide spectrum of consumer products. FPICs, which typically control hardware by functioning either as an amplifier, oscillator, timer, counter, computer memory, or microprocessor, have made it possible to program and revise the logic portion of integrated circuits after manufacture. Depending on the application, FPICs return a series of analog or digital values used to control hardware. FPICs are valued by users, who include hardware programmers in telecommunications, industrial automation, robotics, computing, research, and instrumentation, because they are customizable to a specific application and facilitate adaptability through reprogramming. FPICs can also be repro-

PAUL WATSON

grammed remotely; which greatly reduces network maintenance costs. Due to their portability, FPICs have expanded the population of developers to nonmanufacturers.

The OOPIC programming environment typically involves a microcomputer, the OOPIC device, the controlled hardware device, connecting cables (from the microprocessor to OOPIC and from OOPIC to the target apparatus), and a power cord and source for OOPIC, as illustrated in the figure on the previous page. The OOPIC programming method was created so hardware programmers could capitalize on the inherent strengths of OO programming. OOPIC uniquely binds hardware circuitry with its software representations. Each programmable hardware component has a software object counterpart in the OOPIC programming environment. This includes FPIC-controlled hardware devices as well as the hardware elements contained on the OOPIC Controller. The most common FPIC-controlled hardware devices are supported by the OOPIC programming environment. A full version of the OOPIC Programming Environment software is downloadable for free from www.oopic.com.

Several hardware circuits are integrated into the standard OOPIC Controller: a five-pin programming connector that connects the OOPIC Controller to the PC printer port, a power connector, a 40-pin I/O connector, memory sockets, network connectors, and a prototyping area. Optional peripheral hardware, such as a nonstandard five-volt power cord and source, can be used to adapt the OOPIC architecture to the project. In runtime mode, an OOPIC can be used as a standalone device with the target apparatus, or it can be connected to external hardware components, computers, and other OOPICs using the I2C network protocol created by Philips Semiconductor. Relative advantages of the OOPIC technique over its predecessors are described here.

Architecture	Maximum Instructions Per Second	Maximum Virtual Circuit Operations Per Second
Basic Stamp	10,000	0
OOPIC	2,000	100,000

A performance comparison between OOPIC I and Basic Stamp I architectures.

Ease of Use. The graphical user interface of the OOPIC programming environment is much easier to use than previous methods. For example, encapsulation facilitates information hiding, so users are not presented with irrelevant system properties during application use. By encapsulating programming artifacts, OOPIC uniquely does not require knowledge of electronics principles or hardware mechanics. Encapsulation also allows developers to interface with software objects that look and feel very similar, but that represent real-world hardware objects that are quite different structurally. This functional similarity between structurally dissimilar hardware objects makes hardware replacement much simpler with OOPIC. The GUI aspect of OOPIC has the potential for expanding the population of FPIC programmers to nonengineers, an effect that has implications for changing the structure of the FPIC market and industry.

Ease of Development. Two primary features of the OOPIC development environment simplify development of hardware controls. First, the OO environment allows systems analysts to conceive of complex programming environment artifacts, such as procedures, data, and hardware more logically than its procedural predecessors. Examples of logical procedures supported in the OOPIC object library are conversion functions, clocks, and random number generators. The primary contribution of the OOPIC method is to facilitate traditionally complicated hardware programming with a simple interface. Examples of hardware circuits supported by the OOPIC object library are DC motors, LCD displays, flame detectors, EEPROMs, serial ports, and keypads. Manipulating the physical object is a matter of manipulating a GUI-represented logical object, which is done at runtime by the user, a program, or another software object. Controlling a target hardware device

involves changing the properties of its representative object or writing code to do the same thing at runtime.

Also, the OOPIC platform contains an optional robot control device that greatly simplifies the handling of hardware circuitry when developing robotics applications. OricomTech (www.oricomtech.com) recently released the BOT40, a production-level motor controller board that contains the most commonly used external hardware components used in hardware programming applications. The BOT40 includes an OOPIC controller that can be programmed to control the prescribed devices and more. It allows rapid development by novice programmers, who need not understand even basic electronics to program hardware.

Productivity. The inheritance of object properties within classes and subclasses promises to improve programmer productivity. Remedial steps in OOPIC application development involve selecting program components from an existing, standardized class library. The environment facilitates greater application flexibility by allowing OOPIC programmers to develop user-defined objects and proprietary object libraries. The combined libraries support object reuse and portability in subsequent development projects.

Product Functionality. The propensity to provide greater application functionality and quality is another advantage of OOPIC over its predecessors. The OOPIC method incorporates three features that allow developers to create functionally superior hardware control applications. First, by binding objects, programmers can use the OOPIC environment to create virtual circuits, which help represent object relationships that involve continuous monitoring or updating. Virtual circuits facilitate multitasking by running in the background while the human user interacts with other functions.

Multitasking represents a major product advantage of OOPIC over traditional architectures, such as the Basic Stamp. While the Basic Stamp architecture is faster in performing single tasks, OOPIC represents a paradigm shift in how hardware programmers can work. Savage measures processing speed in the OOPIC by virtual circuit operations (VCO), which are instances of communication between or within objects. The table appearing here illustrates a comparison using both measures for the OOPIC I and Basic Stamp I hardware versions.

Polymorphism allows multiple homogenous messages to be sent to several objects simultaneously. OOPIC uses polymorphism to link several programmable hardware objects together, forming a library of programmable virtual circuits. An object can participate in several virtual circuits at a time, due to the many-to-many relationship between them.

The OO platform facilitates event-driven responding. FPIC applications often respond to real-world events, such as when a nonstationary robot bumps into a wall. When such environmental occurrences trigger changes in object properties, predefined procedures can be called. Procedural events in OOPIC applications can be triggered by any change in a property. OOPIC naturally facilitates event-driven programs by representing real-world events as independent objects. As the application interacts with its environment, the properties of OOPIC's event object change states, which may in turn change the program flow. Creating event-driven applications involves two additional steps during OOPIC development. First, analysts should identify the potential events and associated logic paths that exist in alternative solutions during design. Second, the event object and associated methods should be added to the application during programming.

The OO aspect of OOPIC greatly facilitates a networking orientation. Using the I2C networking protocol developed by Philips Semiconductors, developers can address up to 127 OOPICs in one architecture. Or, developers can link up to 126 coprocessors to a single OOPIC controller. The OOPIC Programming Interface allows developers to manipulate or read from any property on any OOPIC controller on the network. The network can operate autonomously, allowing the controllers to manipulate or read the property values of one another, allowing the network nodes to control one another in a

loosely-coupled environment. The use of the I2C protocol gives OOPIC applications potential connectivity to the Internet and wireless architectures.

Recent Developments

OOPIC developers need to keep abreast of recent developments regarding the methodology. For instance, Magnevation is now developing a light-duty development platform for instructional purposes. Cognitoy (www.cognitoy.com) is developing a platform to explore robotics concepts in virtual reality, facilitating learning about traditionally complex mechanical, electrical, and software aspects of hardware programming. Control Your World Innovations Limited (cywil.ca) is developing a Web-accessible OOPIC controller that allows users to control hardware from a Web-based form. These and other products promise to make the development and use of hardware applications much more palatable for novices. C

References

1. Bourne, K.C. Putting rigor back in RAD. *Database Programming and Design 7*, 8 (Aug. 1994), 25–30.
2. Hsieh, H., Dong, K., Ja, J., Kanazawa, R., Ngo, L., Tinkey, L., Carter, W., and Freeman, R. A 9000-gate user-programmable gate array. In *Proceedings of the IEEE Custom Integrated Circuits Conference* (May 1988).
3. PriceWaterhouseCoopers Technology Center. *Technology Forecast: 2001–2003*. PriceWaterhouseCoopers LLC, Menlo Park, CA.

Gary F. Templeton (templetg@email.uah.edu) is an assistant professor of Management Information Systems at the University of Alabama in Huntsville.

SIGs Announce Candidates for Election

In accordance with ACM Bylaw 6, the following SIGs will hold elections this spring: SIGARCH, SIGCHI, SIGCOMM, SIGDOC, SIGecom, SIGGRAPH, SIGIR, SIGMETRICS, SIGOPS, SIGPLAN, and SIGSAM.

ACM Policy and Procedures require that those SIGs holding elections notify their membership of candidates for elected offices. Below is a list of SIG(s) that have submitted their slate of candidates in addition to those SIGs listed in the February issue of *Communications*.

SIGecom	
Chair	Michael Wellman
	Running Unopposed
Vice Chair	John Riedl
	Daniel Menasçe
Secretary/Treasurer	Amy Greenwald
	Peter Wurman
SIGMETRICS	
Chair	Leana Golubcik
	Scott Leutenegger
Vice Chair	John Lui
	Philip Heidelberger
Secretary/Treasurer	Martin Arlitt
	Robert Berry
Board of Directors	Jennifer Rexford
	Sem Borst
	Jean-Yves LeBoudec
	Margaret Martonosi
	Edmundo de Souza e Silva
SIGOPS	
Chair	Keith Marzullo
	Brian Bershad
Vice Chair	Antony Rowstron
	Gilles Muller
Secretary/Treasurer	Fay Chang
	Andrea Arpaci-Dusseau
SIGPLAN	
Chair	Benjamin G. Zorn
	Michael G. Burke
Vice Chair	Kathleen S. Fisher
	Running Unopposed
Secretary/Treasurer	Jens Knoop
	John Gregory Morrisett
Member-at-Large	Philip Wadler
	Wilson Hseih
	Jack W. Davidson
	Simon L. Peyton-Jones
	Zhong Shao
	David L. Detlefs
	Charles N. Fischer

In accordance with the SIG Bylaws, additional candidates may be placed on the ballot by petition. All candidates must be Professional members of ACM, as well as members of the SIG. Anyone interested in petitioning must inform ACM Headquarters (Pat Ryan, ACM, 1515 Broadway, 17th Fl., N.Y., N.Y. 10036) and the SIG Secretary of his/her intent to petition by March 15th.