

Integrating OO Concepts Into a CS0 Course

Charles Pheatt
Emporia State University
Department of Mathematics and Computer Science
Emporia, KS
620-341-5637
pheattch@emporia.edu

ABSTRACT

This paper describes the use of field programmable integrated circuits (FPIC) in introducing object oriented (OO) programming concepts into a CS0 Course. Using a low cost device known as the OOPIC (Object-Oriented Programmable Integrated Circuits), students can easily control hardware circuitry while being exposed to OO programming concepts. The OOPIC device simplifies hardware programming by allowing students to use common programming languages (Visual Basic, Java, or C) and a simple development environment to create and test programs. In addition to introducing the use of an OO approach to problem solving, the device provides students an opportunity to deal with simple circuits.

Categories and Subject Descriptors

K. [Computers and Education]: Computer & Information Science Education – *Computer Science Education*.

General Terms

Languages.

Keywords

CS0, field programmable integrated circuits, OOPIC.

1. INTRODUCTION

There has been considerable discussion on the approach to be taken (breadth versus depth) as well as numerous proposals for languages that can be used in presenting a CS0 course [5, 7]. Such courses, in general, emphasize problem-solving techniques in addition to building rudimentary programming skills. The Association for Computing Machinery's Computing Curricula 2001 [1] includes requirements incorporated by many such courses.

OO programming is central to the curriculum in most CS programs, and is specifically mentioned in the Computing Curricula 2001, Overview of the CS Body of Knowledge (PL6. Object-oriented programming). Integrating OO thinking and problem solving as early as possible into the curriculum benefits students when taking CS1 and CS2 courses. Early understanding

and integrating OO techniques into problem solving builds a foundation that can be used throughout the computing curriculum.

The author, having taught CS0 courses a number of times and using various texts [2, 4], has incorporated OO concepts as part of the curriculum. However, discussing OO techniques in the abstract or using an artificial environment [2], generally leaves students without a context in which to internalize the paradigm. It was felt that if students could use objects in actual problem solving, their understanding and retention of the concept would be improved. A recent technical opinion article [9] suggested a possible solution to the OO in CS0 quandary. The approach involves using a low cost device known as the OOPIC (Object-Oriented Programmable Integrated Circuits).

Introducing the OOPIC into a CS0 curriculum allows students to experience coding in an event-driven environment beyond those normally encountered. In a traditional or procedural application, the application itself rather than an event controls the portions of code that execute. The exposure to a true OO, event-driven programming environment affords students insights into issues involved in highly interactive or web-based programs.

2. OOPIC

The OOPIC, fabricated by Savage Innovations [8] is a field programmable integrated circuit based on the PIC series of micro-controllers from Microchip Technology. The PIC chip is used in numerous embedded applications and is typically programmed in assembly language or C/C++. The PIC's complexity excludes its use in a CS0 course, but integrated into the OOPIC device, it becomes a tractable platform for novice programmers. It exists in several versions; A.2.x, B.1.x, B.2.x+. The versions are available in several different board configurations, the C, R and S series. Prices range from \$40 to \$90 dollars for the various configurations and versions.

Depending on the version selected, the device is connected to a PC via a parallel or serial port. It is programmed using the OOPIC IDE (available free from Savage Innovations). The environment provides a text editor and debugger and allows the user to select programming language (Visual Basic, Java, or C) and OOPIC version. Once the source code is entered and compiled, it is downloaded to the device via the aforementioned connection. The debugger is rudimentary and requires cable swapping if using the parallel communication option.

The OOPIC has an extensive list of built-in objects that may be used for hardware communication as well as data manipulation. The classifications used are:

- Hardware Objects - these objects encapsulate the functionality of the physical hardware circuits within the OOPIC. They include an analog-to-digital (A2D) converter, 1, 4, 8 and 16-bit I/O, servo control and timer objects.
- Processing Objects - these objects provide various mathematical, logical and other data manipulation functions. They include data conversion, counters, logic-gate functions, mathematical functions (integer only), real-time clock and random number generation.
- Variable Objects - objects that store values. They include manipulators for bit, nibble, byte, word as well as access to random access memory and EEPROM.
- System Objects - an object that allows access to system parameters. Items include reset, pause, timers and voltage source.

The OOPIC versions have a number of common hardware features that are of note:

- Built-in 8 or 10-bit A2D converters, depending on version.
- Memory (72 to 256 bytes) dedicated to the storage of program variables.
- Expandable program code memory (8KB EEPROM).
- Memory dedicated specifically for the storage of objects. This high-speed memory is used for storing pointers to native PIC code implementing the objects. The fetch-execute loop for objects runs approximately 100 times faster than user program code stored in EEPROM. The objects may be formed into virtual circuits. Virtual circuits (VC) appear to be physically discrete electronic circuits, but are actually emulated by the OOPIC operating system.
- A port compatible with the Phillips I2C protocol that may be used to connect I2C devices (A2D's, clock/calendars, temperature/voltage monitors, tone generators, etc.) or to interconnect OOPIC devices.

In addition, the OOPIC allows the use of polymorphism to link several objects together. An object can play a part in several VCs at one time.

```
// declare the LED object
ODio1 LED = New oDio1;
void Main(void)
{
  // LED will be connected to I/O line 31
  LED.IOLine = 31;
  // line will be an output line
  LED.Direction = cvOutput;

  do
  {
    // blink LED once per second
    LED.Value = OOPic.Hz1;
  } while(cvTrue); // loop forever
}
```

Figure 1. The Embedded System "hello world" Program.

3. CREATING OOPIC OBJECTS

The OOPIC may be programmed in Visual Basic, Java, or C. Objects are simply declared with a new statement and subsequently referenced. An example of the standard embedded system "hello world" program [3] is shown in Figure 1. The "hello world" program in embedded systems parlance is setting up the device and associated hardware (in this case a light emitting diode (LED) and resistor), and having the LED blink. Note that the Java/C syntax is used in the example.

Objects may be combined into virtual circuits. Linking together a set of OOPIC objects creates a VC. The objects comprising a VC are evaluated in the fetch/execute loop where they are evaluated in the order in which they were created. Any change of state is applied to the object at evaluation time. Objects are joined using link methods. When a link method is encountered during the program's execution, an object's property, which is given as the argument, is assigned to the base object specified by the link. After the Link method is executed, the base object will use the value of the linked Property for its operation.

Consider the VC shown in Figure 2. In this simple circuit we will take input from single bit objects A and B, perform a logical OR on the inputs with object C, and pass the result to a single bit output object D.

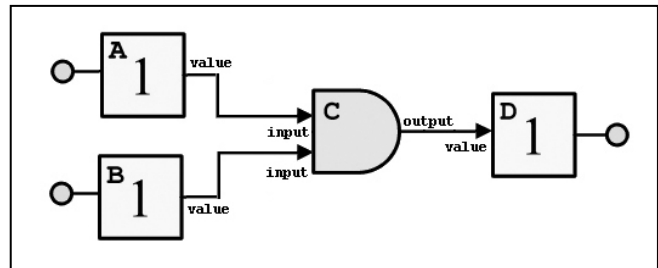


Figure 2. OOPIC Virtual Circuit.

The OOPIC code to implement the VC is shown in Figure 3 using the Visual Basic language option. Note that two input objects, an OR object and an output object are first created using DIM statements. The first portion of the main program defines the output lines associated with each object as well as input/output direction. The second portion of the code uses the OOPIC link syntax to join all the components.

```
Dim A As New oDio1
Dim B As New oDio1
Dim C As New oGate(2)
Dim D As New oDio1
Sub Main()
  A.IOLine = 8
  A.Direction = cvInput
  B.IOLine = 10
  B.Direction = cvInput
  D.IOLine = 12
  D.Direction = cvOutput

  C.Input1.Link(A)
  C.Input2.Link(B)
  C.Output.Link(D)
  C.Operate = cvTrue
End Sub
```

Figure 3. OOPic Virtual Circuit Code.

4. INTEGRATING THE OOPIC INTO CS0

Courses introducing object-oriented computer programming for computer science majors and computer professionals typically include simple data types, control structures, an introduction to array and string data structures and algorithms, debugging techniques and exposure to the history of computing. Such courses typically emphasize object-oriented design, good software engineering principles and the development of fundamental programming skills.

The CS0 course using the OOPIC technology has been broken into three major segments:

1. The first seven weeks of the semester-long course introduces students to programming and problem solving using the Visual Basic (VB) language. The emphasis is on programming structures such as subroutines, functions and control structures. Decidedly de-emphasized are GUI development techniques since, based on previous experience, elaborate program interfaces detract from students acquiring the conceptual programming framework.
2. Once the students have grounding in VB programming, the OOPIC technology is introduced. A total of five weeks is invested in learning the OOPIC IDE and language syntax. Four activities are required to be completed as part of this portion of the course. These activities are discussed in a following section.
3. The final 4 weeks of the semester are devoted to students learning and using POV-Ray. Using POV-Ray exposes students to a C/Java like scene description language that is used to generate ray-traced graphics. As a final project for the course, students are required to generate a 60-second (1440 frame) video. Students in general enjoy this activity. It also provides a key learning experience; that being, "doing computer animation is really difficult and time consuming".

Experience from CS0 offerings has shown that exposure of this type allows students to enter CS1, "hitting the pavement running". Basic program form as well as control structures have been presented and students appear to have an easier time internalizing OO concepts as they are introduced in CS1.

5. OOPIC ACTIVITIES

A number of activities have been developed to allow students to experiment with the OOPIC device. Early activities introduce the device, while later ones emphasize using the device for data acquisition and communicating to a PC by using a VB program. The activities are listed in the order that they are presented to students.

- **Hello world** – to gain familiarity with the OOPIC IDE and the device in general, students implement a single flashing LED program. Implementing this program is also useful for students when trouble shooting the device. After completing "hello world", students then expand the circuit and program to implement 4 LEDs that display binary counts.
- **7-segment LED display** – in this activity students implement the circuit and code that display the digits 0-9 on the 7-segment LED display. This exercise introduces the use of subprograms and functions as well as timing and looping.

Using VCs in this exercise plays an important role in developing a solution with parsimonious coding.

- **Distance measurement** – students use the OOPIC oA2D object which is an 8-bit A2D converter. This is used in conjunction with a Sharp GP2D12 IR or Sharp GP2D120 IR ranging module. The sensor takes a continuous distance reading and reports the distance as an analog voltage with a distance range of 10cm (~4") to 80cm (~30") for the GP2D12 and 4cm (~1.5") to 30cm (~12") for the GP2D120. Students first establish communication between the OOPIC and a VB program using a serial port connection. The students collect voltage versus distance data using the data they have collected and then modify the OOPIC and VB programs so that the PC display shows "on track" when the sensor is between 6 to 10 inches from an obstacle and displays "off track" when the sensor detects a distance outside the range.
- **Capacitor Discharge** – this activity has students utilize a circuit with a 1 Farad capacitor and the OOPIC oA2D object. The circuit and OOPIC program communicate with a VB program in which students accumulate voltage data and generate a plot of the charge and discharge rate of the capacitor. The activity builds on the experience gained in the distance measuring activity. The activity also allows students to gain experience with timing issues associated with data acquisition and presentation as well as generating plots.

Component requirements and sources, schematics and sample code for these activities are available at [6].

6. CONSIDERATIONS

Using the OOPIC in CS0 has presented some unique challenges. Documentation for the device is available at Savage Innovations [8] as well as [3], but neither source is well suited for the novice programmer. Both sources provide some materials that serve as a starting point for activities, but considerable effort is required to provide the level of detail necessary so that a CS0 student can successfully implement them. At this time, there is no other comprehensive source of information on the OOPIC. It is hoped that as additional published materials are made available, the OOPIC will gain wider acceptance.

In the author's use of the device in support of CS0, several OOPIC workstations were set up using OOPIC S and R board styles. The OOPICs are connected to several older PCs available in a general use computer lab. Also provided are several breadboards (for circuit fabrication) and the electronic components necessary for each of the activities. These dedicated stations provide a sufficient resource for the approximately 40 students taking the CS0 course. If OOPIC use was expanded within the course, student purchase of the device might be considered.

7. CONCLUSIONS

The OOPIC has been successfully adapted for use in introducing OO programming concepts in a CS0 course. This course provides an entrée for students into programming constructs and provides exposure to OO problem solving.

Since CS0 using the OOPIC has only been offered twice, it is too early to assess its impact. Student reaction has been very positive, with a number of comments suggesting that the OOPIC provided a “real” context in which to use OO techniques. A number of students have also expressed interest in using embedded technology in other courses in the CS curriculum.

Based on this early experience, the author is continuing to develop additional activities that can be used with the OOPIC. Expanding the amount of time devoted to OOPIC programming in CS0 is also being considered.

8. REFERENCES

- [1] ACM and IEEE, Computing Curriculum 2001 Computer Science, version of December 15, 2001. Available at <http://turing.acm.org/sigs/sigcse/cc2001>.
- [2] Bergin, J., Stehlik, M., Roberts, J., Pattis, R., *Karel++: A Gentle Introduction to the Art of Object-Oriented Programming*, John Wiley & Sons, New York, N.Y., 1996.
- [3] Clark, D., *Programming and Customizing the OOPic Microcontroller: The Official OOPic Handbook*, The McGraw-Hill Companies, Hightstown, NJ, 2003.
- [4] Decker, R., Hirshfield, S., *The Analytical Engine: An Introduction to Computer Science Using the Internet*, Brooks/Cole Publishing Company, Pacific Grove, CA, 1998.
- [5] Denning, P., Comer, D., Gries, D., Mulder, M., Tucker, A., Turner, A., Young, P., “Computing as a discipline: preliminary report of the ACM task force on the core of computer science”, ACM SIGCSE Bulletin , Proceedings of the nineteenth SIGCSE technical symposium on computer science education, Volume 20 Issue 1, February 1988.
- [6] Pheatt, C., OOPIC Activities. Available at <http://pheattch.emporia.edu/oopic>.
- [7] Reed , D., “Rethinking CS0 with JavaScript”, ACM SIGCSE Bulletin , Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education, Volume 33 Issue 1, February 2001.
- [8] Savage Innovations, <http://www.oopic.com>.
- [9] Templeton, G. Object-Oriented Programming of Integrated Circuits, Communications of the ACM ACM, Volume 46, Number 3, March 2003, pp. 105-8.