*Global J. of Engng. Educ., Vol.8, No.2*
*Published in Australia*


# Implementing a Software Teaching Module for the Real-Time Simulation of Digital Logic Circuits*

**Charles Hacker**
*Faculty of Engineering and Information Technology, Griffith University*
*Parklands Drive, Gold Coast, QLD 4215, Australia*

In the article, the author presents the development of a software module called *DigitalSim*, which performs the real-time simulation of digital circuits. This *DigitalSim* module forms an addition to the author's developed *WinLogiLab* software, which is a suite of educational software modules that provides tutorials and testing in digital electronics. From its inception, the *WinLogiLab* software was intended to encompass only digital circuit design, with no plan to cover any real-time simulation of circuits. This was due to the present utilisation of various external programs that already adequately perform this simulation. However, feedback from students, as well as lecture needs, indicated that it would be beneficial to actually include simulation within *WinLogiLab*.

## INTRODUCTION

When teaching the many diverse concepts in digital logic theory and circuit design, the author has found it advantageous to utilise computer-based software for student teaching and self-paced testing. To this end, the author has developed a suite of educational software modules [1-5]. These are collectively called *WinLogiLab*, which currently covers the full teaching curriculum within the author's digital electronics course. This includes the teaching and testing of the following:

- Boolean algebra;
- Truth tables;
- Logic circuit diagrams;
- Logic minimisation techniques;
- Combinatorial logic circuits;
- Sequential logic circuits;
- Finite state machines.

From the inception of the *WinLogiLab* tutorial suite, the software was desired to fill a discovered gap in the available external (commercial and shareware) digital logic software [1]. The deficiency being in the absence of currently available software for the student teaching of how to *design* digital circuits for later analysis and testing. There was thus no desire to replicate existing (commercial or shareware) software.

Consequently, the previously developed *WinLogiLab* software modules covered only the areas of teaching and testing of digital logic circuit *design*. The analysis and testing of such designed circuits was then left to be performed in later laboratories, or be real-time simulated by the use of the many existing external digital circuit simulation software packages. However, continuous feedback from students, and lecture needs, indicated that it would indeed be beneficial to include real-time simulation capabilities within the *WinLogiLab* software suite.

Therefore, this article presents the development of this additional software module, called *DigitalSim*, which simulates the real-time operation of digital circuits.

## EXISTING DIGITAL SIMULATION PACKAGES

As stated, the development of real-time simulation software was initially considered unnecessary within the author's digital electronics course. This was due

*A revised and expanded version of a paper presented at the *4th Global Congress on Engineering Education*, held in Bangkok, Thailand, from 5 to 9 July 2004. This paper was accorded a UICEE Director's Choice award (with another paper) for excellence in engineering education at the Conference.

to the course already utilising various external (commercial and shareware) simulation programs, which quite adequately performed the real-time simulation of digital circuits [6-9].

Specifically, the simulation applications utilised in the author's electronics courses are as follows:

- *Electronics Workbench* [6];
- *WinPSpice* [7];
- *Logic Works* [8];
- *Multimedia Logic* [9].

The *Electronics Workbench* and *WinPSpice* applications are mainly covered in other analog electronics courses at Griffith University, Gold Coast, Australia. Although these applications can perform digital simulations, and are available in student laboratories, they were unsuitable for use in the digital electronics course. This was due to, firstly, the complexity of these applications, which necessitated unwanted extra lecture time in just explaining the user operation of the software. Also, the licensing agreement disallowed the ability to provide these software applications to students for their home use.

The application *Logic Works* was the digital simulation software originally chosen for use in the digital electronics course. This was due to the simple operation of the software, as well as the fact that students could easily (with low cost) purchase the software from the University bookshop.

Yet the above *Logic Works* application was superseded by the *Multimedia Logic* application in the author's digital electronics courses. The *Multimedia Logic* application was chosen because it offered greater functionality than the other applications. Another, and possibly more important reason, was that the application has recently become freeware. The freeware status meant that both the *Multimedia Logic* application, and the author's *WinLogiLab* suite, could be provided to students free-of-charge (on CD-ROM).

## THE REQUIREMENTS FOR THE DESIGNED MODULE

Although the above mentioned software applications perform (in most cases) very adequate real-time simulation, student and lecture needs necessitated the development of a similar *WinLogiLab* compatible real-time simulation software module.

The student course evaluations, plus other student feedback, continually requested the development of a *WinlogiLab* compatible real-time simulation software. The students indicated, firstly, that they would prefer to be able to simulate the operation of their *WinLogiLab* designed logic circuit within the actual *WinLogiLab* suite. This would negate the need to manually transfer their circuits to an external software simulation package, which generally required a duplication of effort in entering the same circuit again into the external software.

Secondly, students preferred to only be required to learn one software package. Since the software modules of the *WinLogiLab* suite have a consistent look, feel and operation, it was much easier for students to utilise a compatible *DigitalSim* module than to learn another external simulation application.

Thirdly, students acknowledged that the various *Help Windows* and *Pop-Up Hints* of the *WinLogiLab* software modules made the *WinLogiLab* software more user-friendly than the external simulation packages.

Apart from these student requests, the author had determined a need for developing simulation software for use during lecture demonstrations. The author would, where possible, make use of the existing simulation software applications for lecture presentations. Where the existing simulation software was inadequate, the author would write small animation applications to present those concepts not possible from the existing simulation packages. Thus, the ever-increasing number of *animated* applications also prompted the author to set about developing an actual real-time simulator that was capable of demonstrating the required lecture concepts.

## IMPLEMENTATION OF NEW CODING TECHNIQUES

One of the major hindrances to the author in developing a simulation module was due to the necessity in developing entirely new coding algorithms. The coding already developed for the existing *WinLogiLab* logic circuit *design* modules, were not useable, nor even directly transferable, to algorithms for real-time simulation.

The main reason for the different coding algorithms was due to the fact that the existing algorithms (for the circuit design modules) required no timing or live update. The existing circuit design modules of *WinLogiLab* utilised Boolean algebra, and circuit minimisation rules, in order to arrive at a set constant digital logic circuit.

The real-time simulation algorithms required new coding structures to take into account that the simulated circuit could be in a constantly changing state. Similarly, the new simulation algorithms needed to take into account that there would be a delay

between when a change in the inputs to a particular logic gate will cause a possible change in output of that gate (that is, have logic gate propagation delays).

For example, assume the circuit in Figure 1 is to be simulated. When the user alters the initial input button (on the left of this circuit), the simulation will be required to show the changes in each of the logic gates *rippling* through the entire circuit, (from position 1 to position 4). However, the existing *WinLogiLab* coding structures were only programmed to determine the final (steady state) condition of the circuit, which did not calculate any intermediate steps.

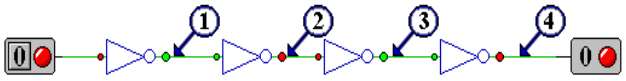


Figure 1: Example of signals *rippling* through a circuit.

The dilemma of providing for a continuously changing circuit, was largely solved by instigating an algorithm that actually implemented a second in-memory circuit, which was identical to the displayed circuit. The displayed circuit was deemed to be the *current state* circuit, and the in-memory circuit was the *next state* circuit. Then, with each time period, the in-memory *next state* circuit was updated based on the values of the other *current state* circuit. At the end of the time period, the *next state* circuit was copied to the *current state* displayed circuit. This process was thus repeated for each time step. As such, this enabled a real-time progressive updating of the displayed circuit.

As mentioned, the underlining algorithm for representing the constantly altering simulation circuit was redesigned for this *DigitalSim* module (compared with other modules). However, apart from this, all of the existing *WinLogiLab* modules coding for graphical interface and user interaction could be essentially reused. This not only saved time and effort, it also had the advantage of giving the developed *DigitalSim* module the same look and feel as the other software modules.

Furthermore, by utilising the same code (for such things as graphical displays, user interfaces and saved data files), the *DigitalSim* module had a consistent user operation to the other modules. In addition, since the save data file format was consistent between modules, the data from other modules could be easily transferred to the *DigitalSim* module.

## THE DESIGNED SOFTWARE MODULE

The *DigitalSim* software module provides a visual real-time simulation of a logic circuit for the user within the main screen of the program. An example of a running simulated logic counter circuit is given in Figure 2.

The *DigitalSim* module allows the user to simulate a circuit that has been transferred (either directly or by loading a saved file) from other *WinLogiLab* modules. Yet the software also allows the user to directly enter their own digital circuit schematic within the *DigitalSim* module itself. This is accomplished by using the toolbar design icons, menu items and the mouse pointer. These are also shown in Figure 2 along the top of the main window.

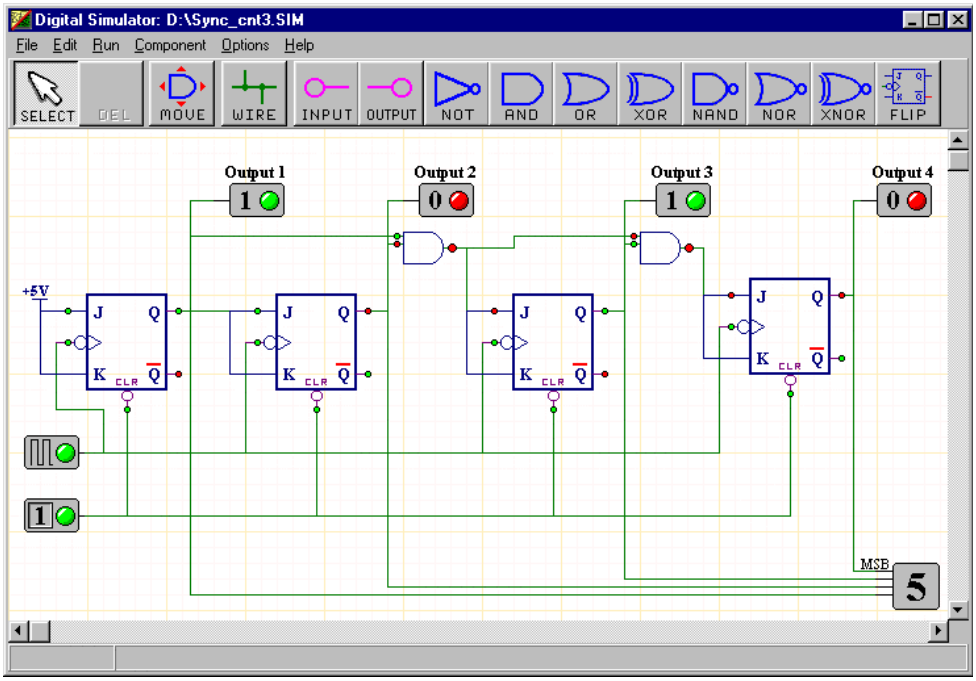


Figure 2: Example of a running simulated binary counter circuit.

## THE SIMULATION START-UP SCREEN

Once a circuit has been inputted, the start simulation button will start the real-time *running* of the circuit. This button is located on the *Start-up Screen*, which constantly *floats* above the main circuit displaying screen. An example of the Start-up screen is given in Figure 3.

Once the simulation has been started, the *start* button caption changes to the words *stop simulation*, and thus the simulation is stopped by the same button.

The Start-up screen also provides for setting the *simulation speed*. The simulation speed allows for the user selection of the *time steps* that occur between updates in the current displayed circuit. The simulation speed can be continually adjusted during the running of the simulation, if so desired.

A notable difference between the developed *DigitalSim* real-time simulator and other external simulators is that the simulation speed (the time steps) are set much slower in the author's *DigitalSim* module. More specifically, due to the adjustable simulation speed, the maximum adjustable speed is set much slower than other external real-time simulators available. This simulation speed was purposely reduced within the *DigitalSim* software so as to better enable students to visually see the step-by-step events that occur within the running circuit.

That is, the other external real-time simulation packages seem to concentrate on running (simulating) as fast as possible. This fast simulation may better mimic the actual fast processing that occurs within actual digital circuits. However, it is not as educational, for this fast simulation rate is too quick for a student to observe, and thus understand, the actual processes going on in the simulated circuit.

The Start-up screen also allows for setting the initial *state* of the simulation, which is the state of the circuit components before simulation occurs. The requirement for an *initial* start-up state is also one of


Figure 3: Example of the Simulation Start-up Screen.

the biggest differences required in the operation between this and previous software modules.

That is, the previous *WinLogiLab* modules were programmed for *design* only, and these design-only algorithms required no *initial* state setting; this is because the design of circuits effectively only presents the *steady-state* condition.

This lack of an *initial* condition is also the reason why the previous design software modules are not able to handle *feedback circuits*. In a feedback circuit, the output of a particular gate is feedback around the circuit, and eventually cycles back to that same gates own input. An example of a feedback circuit is given in Figure 4.

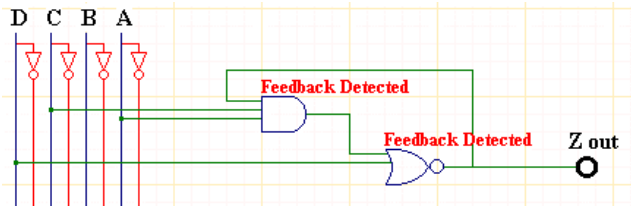


Figure 4: Feedback error in design software modules.

The design-only *WinLogiLab* modules displays an error when the modules encounters a feedback circuit and halts any further processing. This is demonstrated by the *feedback detected* error messages of Figure 4.

The feedback error condition is due to the result of a *chicken and the egg* situation. That is, the output of a gate depends on another gate, which in turn, depends on the first gate, which depends on the other gate, which depends on the first gate again. This cyclic condition would thus continue *ad infinitum*.

In an actual real life circuit, the initial states of logic circuits come up as random (high or low). Although these states are random, each logic device actually has a defined (but unknown) current high or low output. Thus, there is always a set (but random) output on each of the components, even at start-up, in a real digital circuit.

In order to accomplish this effect in the real-time simulation, the circuit must be provided an *initial state* for the logic circuit components. Once this is known, the simulation algorithm can then determine the *next state* that will occur, at the next timing period, given the initial *current state* conditions. The subsequent *current state* is then, as mentioned, copied from the calculated *next state* circuit.

The initial state of the logic circuit components can be set by the Start-up screen as either a logic high (1), or logic low (0), or an unknown random high or low (X). As stated, the unknown random high or low

condition (X) is the more likely event that will occur from a real electronic circuit.

The unknown state (X), does not actually randomly assign a logic high or low. Instead, the logic circuit is assigned the *unknown* (X) state. This unknown state could then ripple through the circuit, like a logic high or low. The unknown state will only be overridden by a logic high or low, when a known high or low condition is set by the user (such as by a switch input). An example of this *unknown* state is given in the early traces shown in the timing window shown in Figure 5. Note that the initial *unknown* states eventually become a logic high or low.

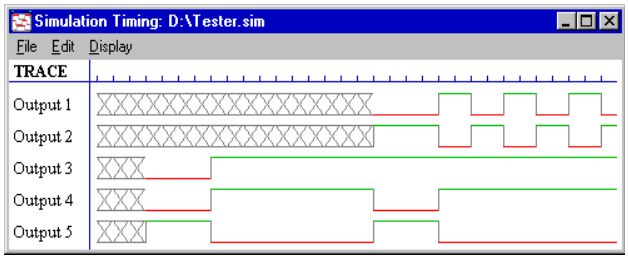


Figure 5: Example of the timing window.

## SETTING INPUTS AND VIEWING OUTPUTS

The real-time simulation software module provides for multiple forms of digital input devices, such as switch inputs and oscillator inputs. This software module also provides multiple forms of output devices, such as output light indicators and hexadecimal output displays. There can be as many input and output devices as the user desires (MS Windows memory space permitting).

During a simulation run, the user can manually modify the logic state of the input components (such as the switches). This is achieved by pressing the buttons available on the input switch component symbols. An example of an input switch component is given in the bottom left hand side of the binary counter circuit exhibited in Figure 2.

In all of the displayed inputs, outputs and logic devices (such as gates and flip-flops), all logic level (high/low/unknown) inputs and outputs to the devices are indicated by simulated (green/red/grey) LED lights. This is extremely useful for always indicating the state of all devices within the circuit. These LED lights continuously display and update while the simulation is running. This feature was lacking from all other researched external (commercial or shareware) real-time logic simulators.

As well as the on-screen simulated LED displays, a timing trace screen also provides a list of the current and past logic levels for all the input and output logic devices. An example of the *Timing Trace Screen* is given in Figure 5.

The on-screen logic levels display as coloured LED lights. However, the timing screen utilises digital square wave pulses. In this case, a peak in the wave represents a logic high (1), the base line represents a logic low (0), and an unknown state (which could be either a logic high or low) is represented by a crossed mark (X).

The timing screens logic square waveform display is similar to the only output display that was available from the researched external (commercial and shareware) logic simulation packages. This conventional output format makes the *DigitalSim* software module's output as effective as the external simulation software. However, as stated, the extra on screen LED displays provides for greater visual status of the circuit's operation.

## ACCESSING EXTERNAL HARDWARE DEVICES

A further addition incorporated into the real-time simulator module was the ability for the software module to access the actual hardware *ports* of the microcomputer. For example, a microcomputer usually contains parallel printer ports, serial communications ports and data ports for analog-to-digital or digital-to-analog converter cards.

From the selection of port module symbols, (as demonstrated in Figure 6), the real-time simulator module can input and output digital signals to the actual physical external hardware port. This enables the *DigitalSim* module to directly control external electronic devices, such as robotic equipment.

An example of the type of external hardware devices developed by the electronic engineering school (at Griffith University) is the *Bilby* robot (shown in Figure 7). This robot is a two wheeled vehicle that is

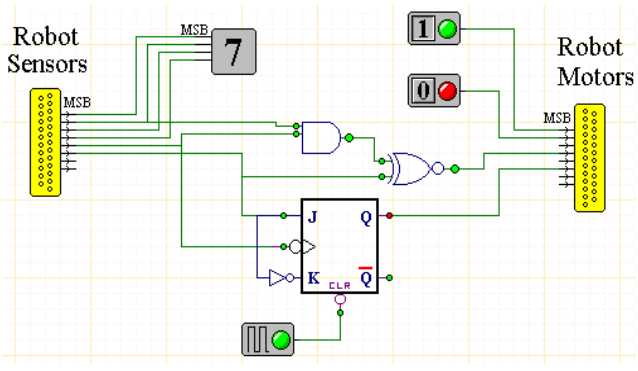


Figure 6: Example of the running *Bilby* robot controller circuit.

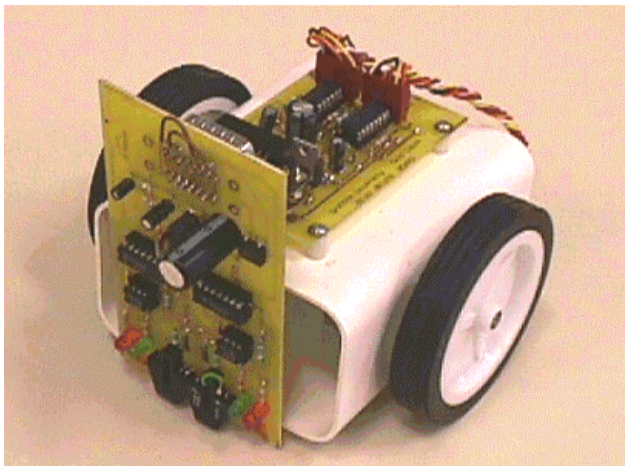

Figure 7: Picture of the *Bilby* robot.

powered by stepper motors, which contain various sensors for track following. The robot is designed to be attached to the printer port of a microcomputer and is thus controlled by commands sent to the printer port by the computer. The robot was initially developed for use as a hardware control device for another course on microprocessor assembly language programming.

However, the robot also happened to be easily controlled by the *DigitalSim* real-time simulator module by utilising Port device components and associated simulated circuitry components. An example of a circuit to control the robot is given in Figure 6.

## CONCLUSION

The aim of the described *DigitalSim* software module was to develop a user friendly graphical-based software teaching module that performs real-time digital logic simulation and that can also integrate with the author's *WinLogiLab* tutorial suite.

This aim has been satisfactorily achieved, in that the software does indeed perform real-time simulation, which integrates with the author's *WinLogiLab* tutorial suite. Also, the animated, visual and graphical interface of the MS Windows environment makes for a user-friendly program.

The *WinLogiLab* module, which included the *DigitalSim* software module, has been successfully incorporated into the curriculum of the author's second year digital electronics course. Student feedback (via student comments and questionnaires) indicated the software suite was educationally useful, beneficial to their studies and that it conveyed the desired information.

Although this *DigitalSim* software module performs similar tasks to existing external software for real-time digital logic simulation, the existing software is not compatible to the author's implementation and successfully trialed *WinLogiLab* software suite.

The *DigitalSim* software was developed and is utilised for both lecture demonstration and students' educational use. Students are thus provided the software, which allows them to make use of this software privately and at their own leisure.

The full *WinLogiLab* software has also been made freely available for educational use to the wider community. The complete software is available for download from the Web site of the School of Engineering at Griffith University. The direct URL link to the *WinLogiLab* software Web site is: http://www.gu.edu.au/school/eng/mmt/WinLLab.html

## REFERENCES


1. Hacker, C. and Sitte, R., Development of a computer program, to electronically design digital logic circuits using Boolean algebra. *Proc. 9th Annual Australasian Assoc. for Engng. Educ. (AaeE97)*, Victoria, Australia, 353-357 (1997).
2. Hacker, C. and Sitte, R., A computer based tutorial for demonstrating the solving of digital electronic circuits. *Proc. 10th Annual Australasian Assoc. for Engng. Educ. (AaeE98)*, Queensland, Australia, 509-519 (1998).
3. Hacker, C. and Sitte, R., Implementing the 'Espresso - two level logic minimiser' algorithm in the MS-Windows environment. Proc. 2nd Asia-Pacific Forum on Engng. and Technology Educ., Sydney, Australia, 124-127 (1999).
4. Hacker, C. and Sitte, R., A computer based teaching program for the design of digital counter circuits. *Proc. 3rd UICEE Annual Conf. on Engng. Educ.*, 225-228 (2000).
5. Hacker, C., Computer-based software for testing students in digital logic theory and design. *World Trans. on Engng. and Technology Educ.*, 2, **2**, 281-284 (2003).
6. Interactive Image Technologies, Electronics Workbench 3.0E, Distributed by Applied Electro Systems Australia (1993), http://www.interactiv.com
7. Orcad/MicroSim Corporation, WinPSpice Design Laboratory, Mixed Analog Digital Mode Simulator – Version 9 (Student Version) (1999), http://www.orcad. com, (1999).
8. Capilano Computing Systems, Logic Works Version 3.0. California: Benjamin Cummings Publishing Co. (1995).
9. Softronics Incorporated., MMLogic: A MultiMedia Digital Logic Design System - Version 1.2c, (Freeware) (1997), http://www.softronix.com/logic.html, (1997).

## BIOGRAPHY

Charles Hacker obtained his electrical engineering and applied science qualifications at the University of Central Queensland (UCQ) in 1989. Mr Hacker then started working in the UCQ Physics Department as a demonstrator, tutor and sessional lecturer. In 1991, he obtained a position as a lecturer in electronics in the School of Engineering at Griffith University, Gold Coast, Australia. Mr Hacker went on to complete a Graduate Diploma in Medical Physics at the Queensland University of Technology (QUT) in 1995, and then completed an MPhil in Electronic Engineering at Griffith University in 2002.

Mr Hacker's research is mainly in the area of engineering education, specifically in developing computer-based educational software. His teaching areas include electronics, microprocessors, computer programming and physics.

*This form is also available on the Web at*
http://www.eng.monash.edu.au/uicee/member/MembershipForm.html

## UNESCO INTERNATIONAL CENTRE FOR ENGINEERING EDUCATION
## UICEE
## *MEMBERSHIP FORM - 2005*

Yes, I/we would like to become a member of the UICEE. Please register me/us as:

| | | |
|---|---|---|
| i. | Partner (industrial or academic) ($A10,000 p.a.) | |
| ii. | Sponsor (A$5,000 p.a.) | |
| iii. | Supporter (A$2,000 p.a.) | |
| iv. | Contributing Member (A$500 p.a.) | |
| v. | Individual Member (A$100 p.a.) | |
| vi | Library Subscription (multiple readers) (A$200 p.a.) | |

(i-iv) Institution /Company Name: .....

(i-v) Individual/Contact Surname: .....

First Name: ..... Title: ..... Position: .....

University/Company Address: .....

.....

Country: ..... Postcode: .....

Phone (B): ..... (H): .....

Fax: ..... E-mail: .....

### *Method of Payment:*

| | |
|---|---|
| | Cheque |

for $..... made payable to: **Monash University - UICEE**

| | | | | | |
|---|---|---|---|---|---|
| | Visa | | Mastercard | | Bankcard |

Card Number: __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __

Cardholder s Name: .....

Expiry Date: __ __ / __ __ Signature: .....

| | |
|---|---|
| | Electronic Funds Transfers (EFT) |
| *BSB* | 033 289 |
| *Bank Account Number* | 630 759 |
| *Name of Bank* | WESTPAC - Monash University |
| *Address of Bank* | Melbourne, VIC 3800, Australia |
| ***Please fax us a copy of the EFT for our record.*** | |

**Please copy this form and return to:**

UICEE, Building 70, Monash University, Wellington Rd, Melbourne, VIC 3800, Australia
Tel: +61 3 990-54977, Fax: +61 3 990-51547, E-mail: uicee@eng.monash.edu.au

*Visit the UICEE Web-site at: http://www.eng.monash.edu.au/uicee/*