CDIO Concepts for Computer Engineering Teaching

Yinan Kong
Macquarie University, Sydney, Australia
ykong@science.mq.edu.au

Abstract: *CDIO (conceive-design-implement-operate) concepts have been applied to several courses of study supporting the Computer Engineering degree at the authors' university. The paper describes how CDIO concepts have been applied to two key projects: the development of a traffic-light controller for a complex traffic intersection and a simple bus-structured computer. An important feature of the approach is the identification and documentation of a number of issues which students should consider in each of the conceive, design, implement and operate stages. This has proved successful for the application of CDIO concepts. Examples of the documentation are given.*

Introduction

Engineering education has evolved into the teaching of engineering science where teaching engineering practice was increasingly de-emphasized. Industry in recent years has found that graduating students, while technically adept, lack many abilities required in real-world engineering situations. Hence, major companies created lists of abilities they wanted their engineers to possess (e.g. Boeing's Desired Attributes of an Engineer).

A CDIO (conceive-design-implement-operate) program is based on the principle that product and system lifecycle development and deployment are the appropriate context for engineering education. The CDIO concepts, which basically are a model of the entire product cycle, form a worldwide initiative in Engineering Education. CDIO is considered the context for engineering education in that it is the cultural framework, or environment, in which technical knowledge and other skills are taught, practiced and learned.

The Australasian Association for Engineering Education is an Affiliated Organisation of the CDIO Initiative (CDIO, 2009), and numerous engineering disciplines/departments/schools/faculties worldwide have taken steps to incorporate CDIO concepts in their teaching. At the authors' university, CDIO concepts have been introduced to two new courses of study on Programmable Logic Design and Digital Systems Design. In the former, CDIO concepts have been used in a team project to develop a traffic-light controller for a complex traffic intersection (Wong, Imrie and Xie 2008) and, in the latter, the concepts have been used in a team project to develop a simple bus-structured computer. The documentation for these projects identifies issues for each of the conceive, design, implement and operate stages of the CDIO process.

Design and Build Projects for Computer Engineering Teaching

Programmable Logic Devices (PLDs) such as the Xilinx range of Field Programmable Gate Arrays (FPGAs) (Xilinx 2009) provide educators with a huge scope for the definition of design and build projects in the field of computer engineering. There are many examples of these projects which include a traffic-light controller, arithmetic logic units (ALUs) for computers, floating-point arithmetic units, frequency meters, boundary-scan testers and simple computers. Some of these support a problem-based learning (PBL) approach to teaching.

The wide range of available microcontrollers also allows a huge scope in the definition of microcontroller-based projects. Many of these projects include aspects of data acquisition (e.g. event recorders) and control (e.g. temperature control). The outstanding example of projects that include

aspects of robotics is the micro-mouse competition (Micromouse 2009) which has provided the focus of combined developmental work on sensors, actuators and intelligent controllers.

Traffic-Light Controller

For the inaugural delivery of Programmable Logic Design in 2008, a problem-based learning approach was used where student teams were required to develop a digital controller to control the traffic flow of a complex traffic intersection. The problem is an identifiable, real-world problem, and there is a huge scope for ingenuity in each of the conceive, design, implement and operate stages of the problem solution. A diagram of the traffic intersection is shown in Figure 1.

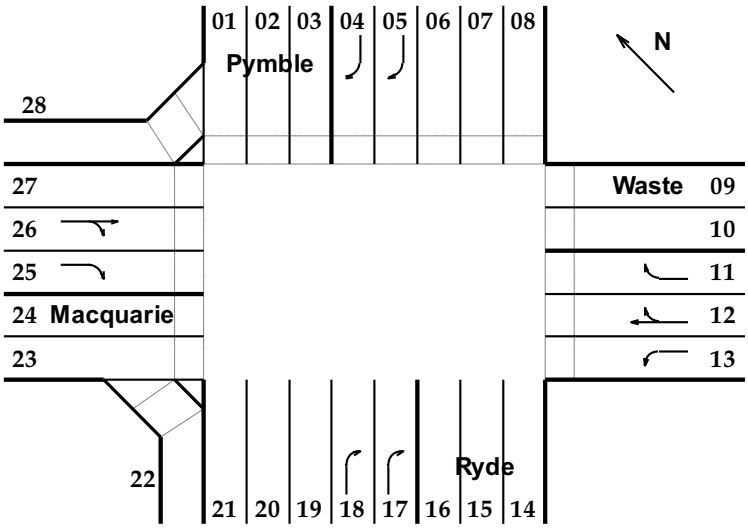

Figure 1: Traffic Intersection

At this intersection, there are ten sets of red, amber and green lights for vehicular traffic, and all of the 16 lanes for traffic approaching the intersection could be installed with traffic sensors. There are other lights and sensors for pedestrian traffic.

This is an 8-week project where students worked 3-4 hours each week on it in groups of 4. The enrolments of the Programmable Logic Design course were 30 and 40 in 2008 and 2009 respectively.

CDIO Stages for a Traffic-Light Controller

The documentation of the project identifies about 30 problems classified under the conceive, design, implement and operate issues of the project. Examples of the 30 problems of each of the conceive, design, implement and operate issues are listed in Table 1.

Table 1: Examples of CDIO Issues for a Traffic Light Controller

CDIO Stage	Problem Area
Conceive	Inputs and Outputs for Digital Control
Conceive	Simplification of Controller Specifications for preliminary development in order to establish the design procedure before the final design is initiated
Conceive	Control Algorithms
Design	Hierarchical Design

Design	FSM (Finite State Machine) State Coding
Design	One-Hot Encoding
Design	Synchronous Delay Circuit
Design	Design for Testability
Implement	GAL/FPGA
Implement	VHDL
Operate	Testing
Operate	Real-time Operation

In the first stage, the customer needs are defined and the technology to be used is selected from the two technologies taught in this course: Generic Logic Arrays (GALs) and Field Programmable Gate Arrays (FPGAs). The detailed specification of the controller is considered a "conceive" issue, and the documentation points out that optimisation of the controller design may be achieved at the initial systems-specification stage of the project.

The design stage of the project is, by far, the most difficult, and about 20 issues are documented. The issues of "hierarchical design", "one-hot encoding" and "synchronous delay circuit" were central to an effective approach to design. After a period of about two weeks into the seven-week period for the project, the teams were alerted to this possibility. In 2008, most of the teams took this "hint", and were able to complete the project on time.

Take the issue of hierarchical design as an example. Six traffic flows of the intersection are usually identified at the top hierarchical level of design as shown in Figure 2. Each flow has its own operating features which turn out to be the second level of design, and each operation has its own combination of states of traffic lights to be designed at the bottom level.

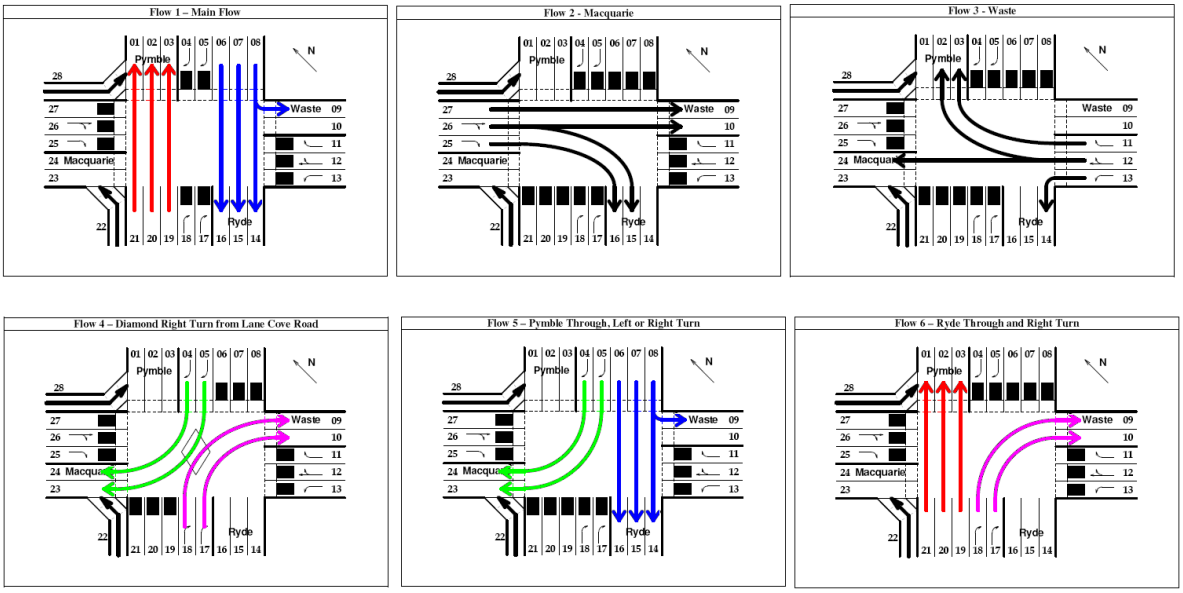

Figure 2: Six Traffic Flows at the top Level Design of the Traffic Intersection

Hardware and software resources of our laboratory support several options for the implement stage. The hardware implementation tools using GAL and FPGA technology are illustrated in Figure 3 and Figure 4 respectively.

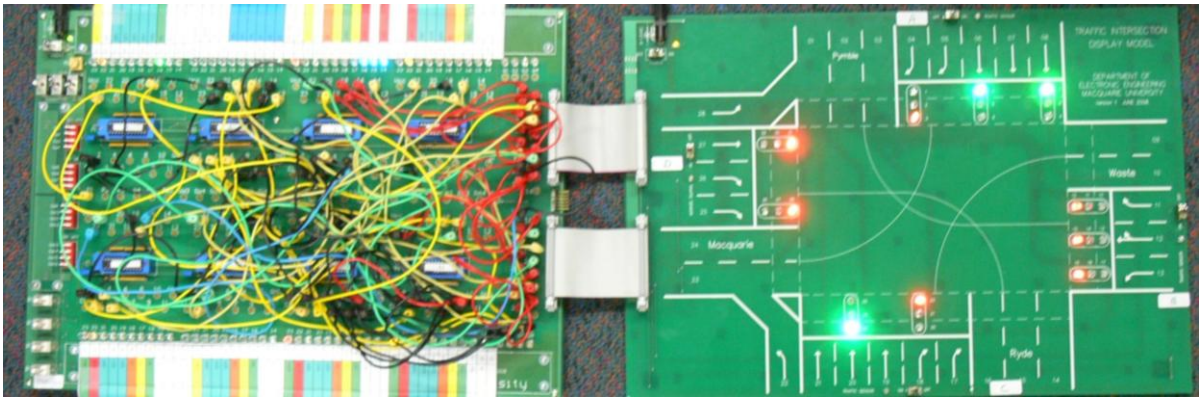

Figure 3: GAL Implementation of the Traffic Controller

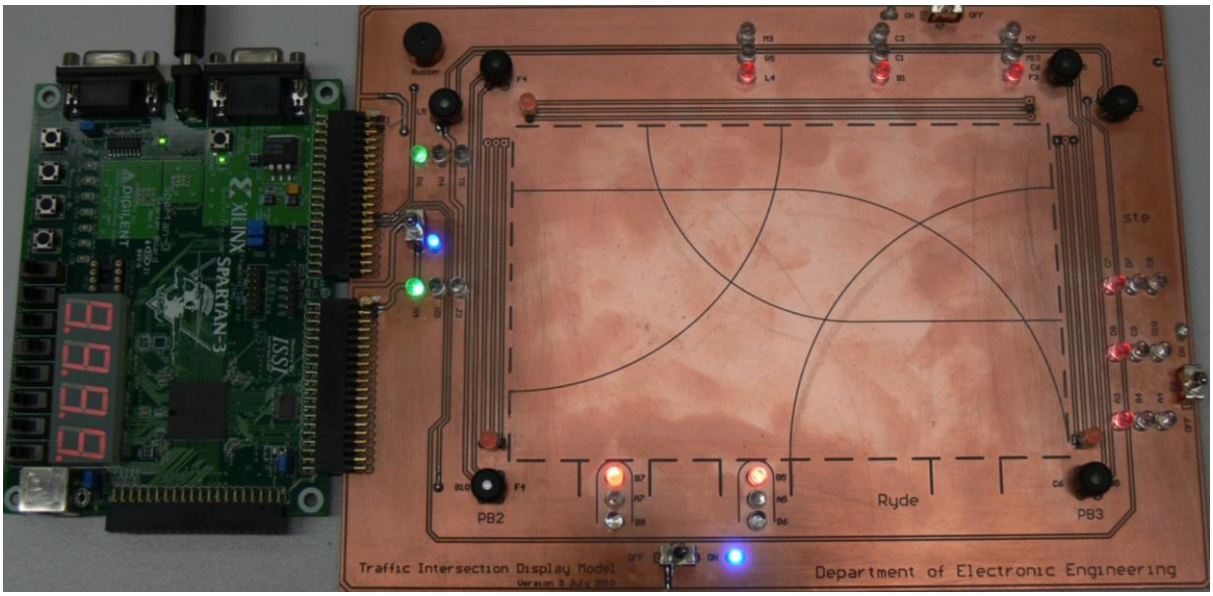

Figure 4: FPGA Implementation of the Traffic Controller

The operate stage also uses the hardware models of the intersection shown in Figure 3 and Figure 4.

Bus-structured Computer

Another course on Digital Systems Design had its inaugural delivery in Semester 2, 2009. 8 students were enrolled for this course where the major project was the development of a simple computer using programmable logic devices (GALs or FPGA). The final documentation will also identify conceive, design, implement and operate issues of the project. A block diagram of the computer is given in Figure 5.

The computer is modelled on the architecture of the 68HC11 microcontroller (Spasov, 2004) which is studied in an earlier digital course. Operands are represented to only 4-bit accuracy and memory size is only 256 words. The instruction set contains only 16 instructions, but two condition codes and several addressing modes are implemented.

While the documentation of the project describes a simplistic computer, expansion of the computer's functionality may be effected in many ways. An expansion of word size from 4 bits to 8 bits (corresponding to that of the 68HC11) may be readily achieved with a FPGA implementation. This will allow a huge increase in the number of instructions of this CISC (Complex Instruction Set

Computer). More programmer-accessible registers, including accumulators and index registers, is a simple extension. A stack pointer would support subroutines and interrupts. Introduction of input/output ports would provide a huge increase in capability.

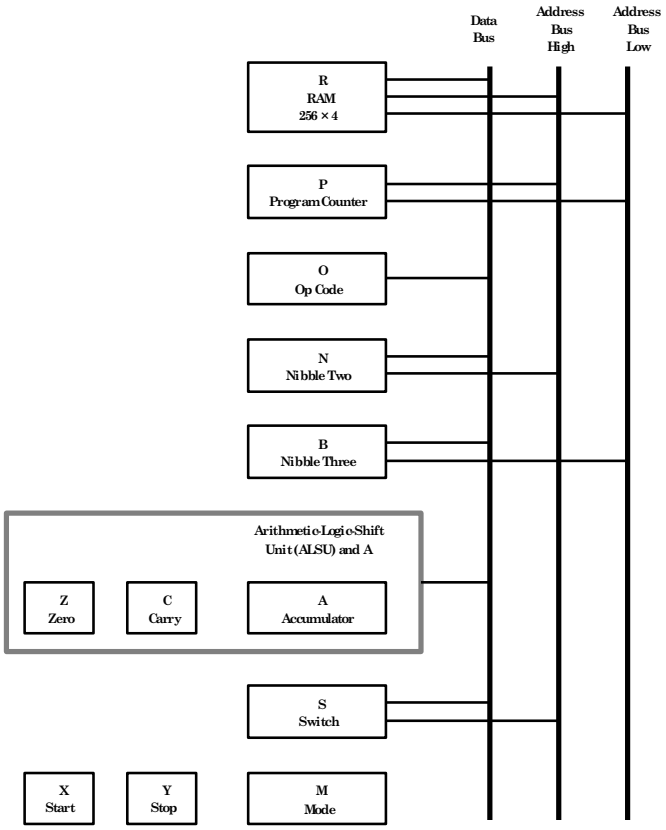

Figure 5: Block Diagram of Bus-structured Computer

CDIO Stages for a Bus-structured Computer

Documentation of the project includes CDIO issues. Some examples are given in Table 2. Each of the CDIO issues is supported by notes in a standard format. An example is given in Table 3.

Table 2: Examples of CDIO Issues for a Bus-structured Computer

CDIO Stage	Problem Area
Design	Hierarchical Design
Design	Expandable Design
Design	Bus
Design	Memory
Implement	GAL/FPGA
Implement	VHDL
Operate	Real-time Operation
Operate	Computer Programs for Testing

Table 3: Example of the Documentation of a CDIO Issue for a Bus-structured Computer

Design Problem Area	Hierarchical Design
Problem	A hierarchical design approach is invariably used for the design of complex systems. The problem is how to partition the system so that it may be represented by a multi-level structure of blocks that support "top-down" specification and "bottom-up" implementation.
Proposed problem-solving approach	At all levels of the hierarchy, focus on inputs to blocks and outputs from blocks. Identify the functionality of important signals interconnecting blocks. Attempt to use some blocks a number of times. Try to "parameterise" blocks (e.g. a MOD-n counter with n as a parameter). Build "expandable" blocks (such as "cascadable" counters).
Notes/Solution	Some advantages of hierarchy in Digital Systems Design are: • modular (division of the system into functional blocks) • multi-level (ability to specify modules in terms of library components or previously defined modules) • simplifies system specification (as details of each module are defined only once), and eliminates repetitious detail • highlights the interconnections between (functional) modules (i.e. highlights the functionality of signals that interconnect the modules) You may wish to add other advantages to the list. Make notes on how you will proceed with the hierarchical design approach.

Conclusions

CDIO concepts have been applied successfully to several Computer Engineering courses at the authors' university. This paper has provided examples of the issues for two team-based projects. A standard format (in an A4-size sheet) has been used for the documentation of each issue. This documentation is included in the notes that are available at the beginning of the semester before the start of the team project. Examples of the documentation have been provided.

The documentation gives students some background or insight to the CDIO issues and suggests meaningful activities. Class surveys have given encouraging, supportive feedback and we are of the view that the application of CDIO concepts to team-based projects is an effective approach to engineering education. Our initial success with the CDIO approach has strengthened our view that the approach has sufficient merit to warrant consideration when any engineering curriculum review takes place.

References

CDIO (2009). *CDIO Initiative.* Accessed at http://www.cdio.org/index.html/ on 10 July 2009

Wong, Imrie & Xie (2008). *Problem Based Learning Applied to a New Unit of Study on Programmable Logic Design, AaeE Conference Proceedings,* ***Rockhampton, December 2008***

Xilinx (2009). *Xilinx.* Accessed at http://www.xilinx.com/ on 10 July 2009

Micromouse (2009). *Micromouse History.* Accessed at http://www.micromouse.cannock.ac.uk/history.htm/ on 10 July 2009

Spasov(2004). *Microcontroller Technology – The 68HC11 and 68HC12, 5th ed.,* ***Pearson***

Acknowledgements

The author would like to thank Professor Anthony Parker, Head of the Department of Electronic Engineering, for his support of the work described in this paper.

Copyright statement