TEACHING STATE MACHINE DESIGN THROUGH A PRACTICAL HANDS-ON EXPERIENCE

Wayne Lu

Department of Electrical Engineering and Computer Science
University of Portland
Portland, OR 97203

Abstract

State machines are the central control unit of autonomous digital systems much like that of the microprocessors to the personal computers, therefore, state machine design has been a major topic covered in every digital design course. Most textbook examples used to illustrate state machine design involve a single state machine, whereas practical applications normally require multiple state machines working together in a synchronized order. To fill this gap, this paper presents a design example involving multiple state machines working as a system. The most unique feature is that this design is an all-digital autonomous system which can be easily implemented by seven PLDs (programmable logic devices) and fitted on a mid-sized breadboard. These PLDs can be easily consolidated into a single CPLD (complex PLD). By implementing the design, students not only can learn how a state machine works, but also can gain hands-on digital systems design and debugging skills. They will also gain experience in consolidating a multi-chip design into a single device.

Introduction

This paper presents an interesting example for teaching practical state machine design. State machines provide control functions for autonomous digital systems much like that of the microprocessor to a personal computer, therefore, state machine design has been a major topic covered in every digital design course. Normally, the state machine design is taught by first introducing the Moore and Mealy machine structures and then discussing the design procedure using examples ranging from simple counters, automobile tail lights flashing, to sophisticated vending machines [1], [2], [3]. These design examples are straightforward and valuable for illustrating basic state machine concept and design procedure. However, these examples consist of only a single state machine. Whereas in practical applications such as MP3 player design [4], the main control state machine consists of many lower-level supporting state machines. To help students learn how multiple state machines can work together in a synchronized order, this paper presents a simplified 4-story elevator controller design example consisting of three state machines.

Although the design task might sound very difficult, however the entire system can be implemented by seven PLDs (programmable logic devices) and fitted on a mid-sized breadboard. Designing and implementing such a system not only can help students learn how a practical state machine works, but also provides an opportunity to exercise the top-down digital system design methodology which is to systematically decompose a complex design into simpler functional units. The limited resource of the popular PLDs such as GAL16V8, GAL20V8, and GAL22V10 also imposes a design constraint in implementing a complex design so that students can learn how to manage the device resources.

This design example also is an excellent example for teaching how to consolidate a multi-chip design into a single-chip design using a CPLD (complex PLD). The procedure of merging these seven ABEL programs into a single program targeted for an XILINX XC9572 CPLD is presented. The program can also be easily converted into the popular Verilog or VHDL

hardware description language. To gain more advanced design experience, the 4-story elevator controller can be extended to handle more floors and/or multiple sets of elevator cars.

Operations Of The 4-Story Elevator Controller

This 4-story elevator design example consists of three state machines and multiple control logic circuits to handle service requests, elevator motion controls, doors opening/pausing/closing operations, floor arrival signal generations, and elevator status displays in an autonomous and synchronized order.

The elevator system consists of a wall-mounted call button on each floor and six buttons inside the elevator: four buttons marked as 1, 2, 3, 4 for specifying the destination floor, an OPEN button for opening the doors, and a CLOSE button for closing the doors. The display inside the elevator includes a 7-segment display showing which floor the elevator is currently on and four red LEDs showing the up and down arrows. While the elevator is moving toward a destination floor, the 7-segment display flashes the next arriving floor.

The purpose of the controller is to interface with the buttons, to control the elevator operations, and to display the status of the elevator. The elevator travels between the first and fourth floors. When arriving at a floor, the elevator:

1. opens its doors (this doors' opening movement takes 4 seconds),
2. pauses for 8 seconds (when the OPEN button inside the elevator is pressed, the door will be kept open), then
3. the doors close automatically (the doors' closing movement takes 4 seconds).

If a wall-mounted button on a floor is pushed, an LED next to it is lit up and stays lit until the elevator arrives at the floor. It takes 8 seconds to travel from one floor to the next floor. A counting state machine keeps track of the elevator travel time and informs the controller whenever the elevator has arrived at a floor. When the elevator arrives at a requesting floor to discharge passengers, it opens its doors, pauses to let the passengers in and out, closes its doors, and moves toward the next requesting floor. If there is no requesting floor, the elevator rests at the floor it last stopped. If there is a call for service later, the elevator then moves toward the requesting floor. If the requesting call is from the floor that the elevator is resting, the elevator simply opens its doors.

To make this design example an all-digital autonomous system, the mechanical movements such as the doors closing and opening movements and the sensors for detecting the elevator's arrival are simulated by digital logic. The doors' movements are simulated by an HLMP-2350 4-LED bar: when all the LEDs are on, the doors are closed; when all the LEDs are off, the doors are open; when the doors are closing or opening, one LED is turned on/off on each side of the LED bar at a time to simulate the doors' movement. The elevator's traveling is simulated by two cascaded HLMP-2350 forming an 8-LED bars: one LED is turned on sequentially at a time in an upward or downward pattern for a second to simulate the elevator's movement between two adjacent floors. A 1 Hz clock is used to control the elevator's operations such as opening the doors, keeping the doors open, closing the doors, and traveling between floors.

Implementation Of The Elevator Controller

The wall-mounted buttons and the buttons inside the elevator are implemented by normally-open, momentarily-on pushbuttons. Each button, except OPEN and CLOSE, has an accompanying red LED which lights up when a call is pending. The elevator controller described above can be implemented by seven PLDs as shown in the system functional block diagram (Fig. 1). The function of each PLD is detailed below:

- U1 (GAL16V8/PALCE16V8) for interfacing with the wall-mounted buttons.
- U2 (GAL16V8) for interfacing with the

buttons inside the elevator except the OPEN button.

- U3 (GAL22V10/PALCE22V10) for the elevator main control state machine and a timing control state machine.
- U4 (GAL20V8) for the doors control state machine and arrival signals.
- U5 (GAL16V8) for simulating elevator travel movements between floors.
- U6 (GAL16V8) for controlling the 7-segment display.
- U7 (GAL16V8) for controlling the up and down arrows inside the elevator.

ABEL [1] is a very easy to learn language and students can comfortably learn ABEL programming within a few hours. The ABEL source files for each PLD can be downloaded from the author's website at www.egr.up.edu/contribu/lu. The theory of operations for each ABEL source file is described below.

- Wallbtns.abl (U1) implements four SR-latches for latching the wall-mounted buttons. Each time a button is pressed, it asserts the Q output which flags a request to the elevator control state machine and also turns on its accompanying red LED. The red LED stays lit until resets by the floor arrival signal.

- Ebuttons.abl (U2) also implements four SR-latches for latching the floor buttons inside the elevator. When a button is pressed, it flags a request to the elevator control state machine and turns on its accompanying red LED. The red LED stays lit until resets by the floor arrival signal. These red LEDs are independent of the floor LEDs.

- Mainctl.abl (U3) implements the main control state machine and an 8-second timing control state machine. The main control state machine consists of six D flip-flops that are divided into three groups of two bits each. One group controls the doors' movements: closing, opening, closed, or open. The second group controls the elevator's moving up or down direction. The third group indicates the floor number. The state machine consists of 22 states defining the doors opening, doors open, doors closing, and doors closed states at

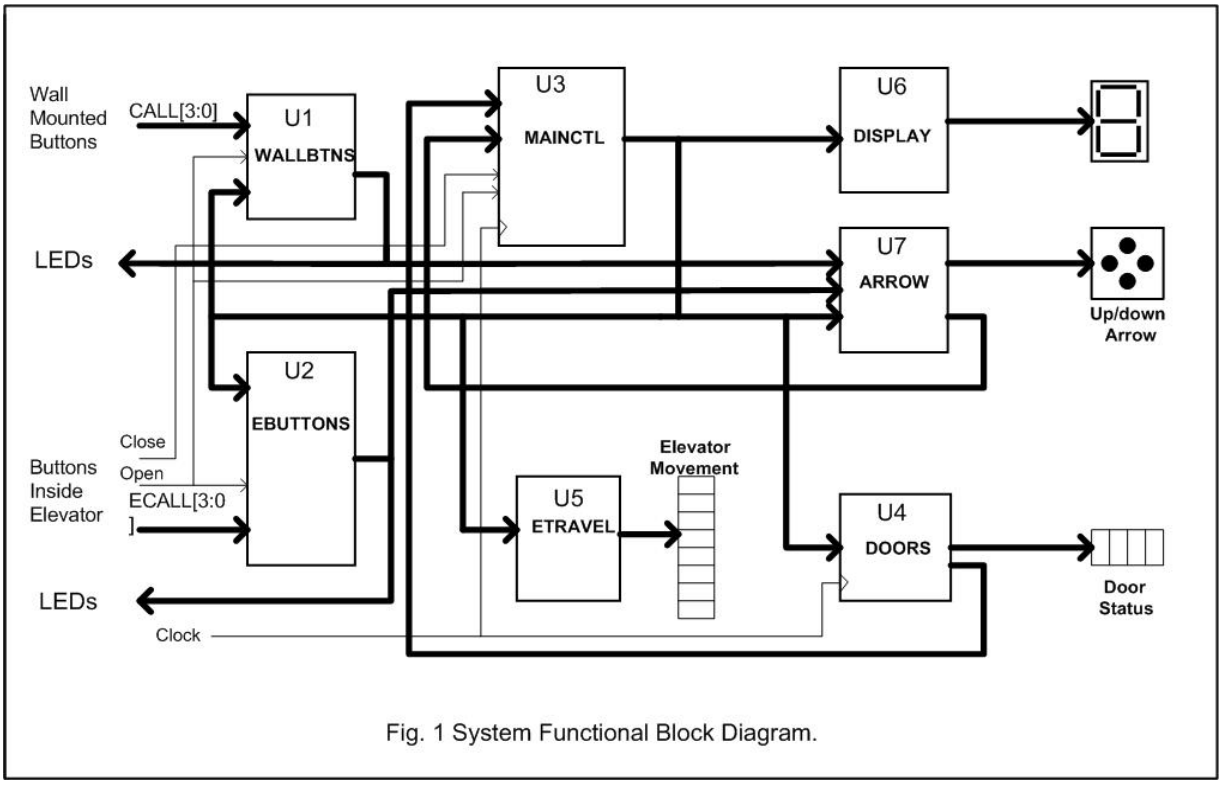

Fig. 1 System Functional Block Diagram.

each floor. The requests from the wall-mounted buttons, buttons inside the elevator, and elevator's arrival signals effect the elevator's operations. If there are multiple pending requests, the state machine will stop at each requesting floor sequentially until the highest requesting floor is serviced. If there are requests from the lower floors just serviced, the elevator will continue its trip to the highest floor before traveling down. The 8-second timing control state machine consists of three D flip-flops for timing the doors' opening and closing movements, elevator travel movements, and the open call button inside the elevator. The controller state machine is reset to the initial state of resting at the first floor by a power-on RC reset circuit.

- Doors.abl (U4) implements the elevator doors control state machine and arrival signal generation unit. The elevator doors control state machine consists of four D flip-flops for simulating the doors closing and opening mechanical movements. The arrival signal generation unit provides arrival signals to the main control state machine to trigger state transitions.

- Etravel.abl (U5) implements the logic of simulating elevator mechanical travel movements and moving direction.

- Display.abl (U6) controls the 7-segment displays. The floor number flashes the next arriving floor number when the elevator is moving toward the destination floor and displays a steady floor number when the elevator stops at a floor.

- Arrow.abl (U7) controls the up and down arrows when the elevator is moving. No arrow is displayed when the elevator is not moving. It also combines the corresponding floor requests from wall-mounted and in-elevator buttons into a single floor request to the main control state machine.

The entire system consisting of the elevator controller and user interface can be fitted on a 6.9" x 5.8" breadboard as shown in Fig. 2 and the detailed schematics are shown in Fig. 3.

Implementing the design using multiple components not only allows students to individually field test each state machine and subsystem, but also lets them experience the process of porting a multi-chip design to a more complicated device architecture.

Consolidating Multiple PLDS Into a Single CPLD

The above seven ABEL programs and PLDs can be easily consolidated into a single program targeted for a single CPLD in less than an hour by some simple cut-and-paste editing operations as detailed in the following steps.

1. Copy all the seven files into a single file.
2. Keep the first module, title, and the last end statements. Delete all the other intermediate module, title, and end statements.
3. Move all input signals from each earlier PLD to the beginning of the file and delete their pin numbers.
4. Move all output signals from each earlier PLD after the input signals and delete their pin numbers.
5. Comment out the input signals which are the outputs from an earlier PLD.
6. Relocate all the intermediate equations from each earlier PLD after the output signals in an order based on the signal dependency.
7. Rename conflicted signal names such as state definitions and intermediate signals.
8. Define intermediate equations to link different connected input and output signals.
9. Relocate the equations and state- diagram statements in an order based on signal dependency.
10. Compile the new consolidated program and fix any overlooked signal names or order.

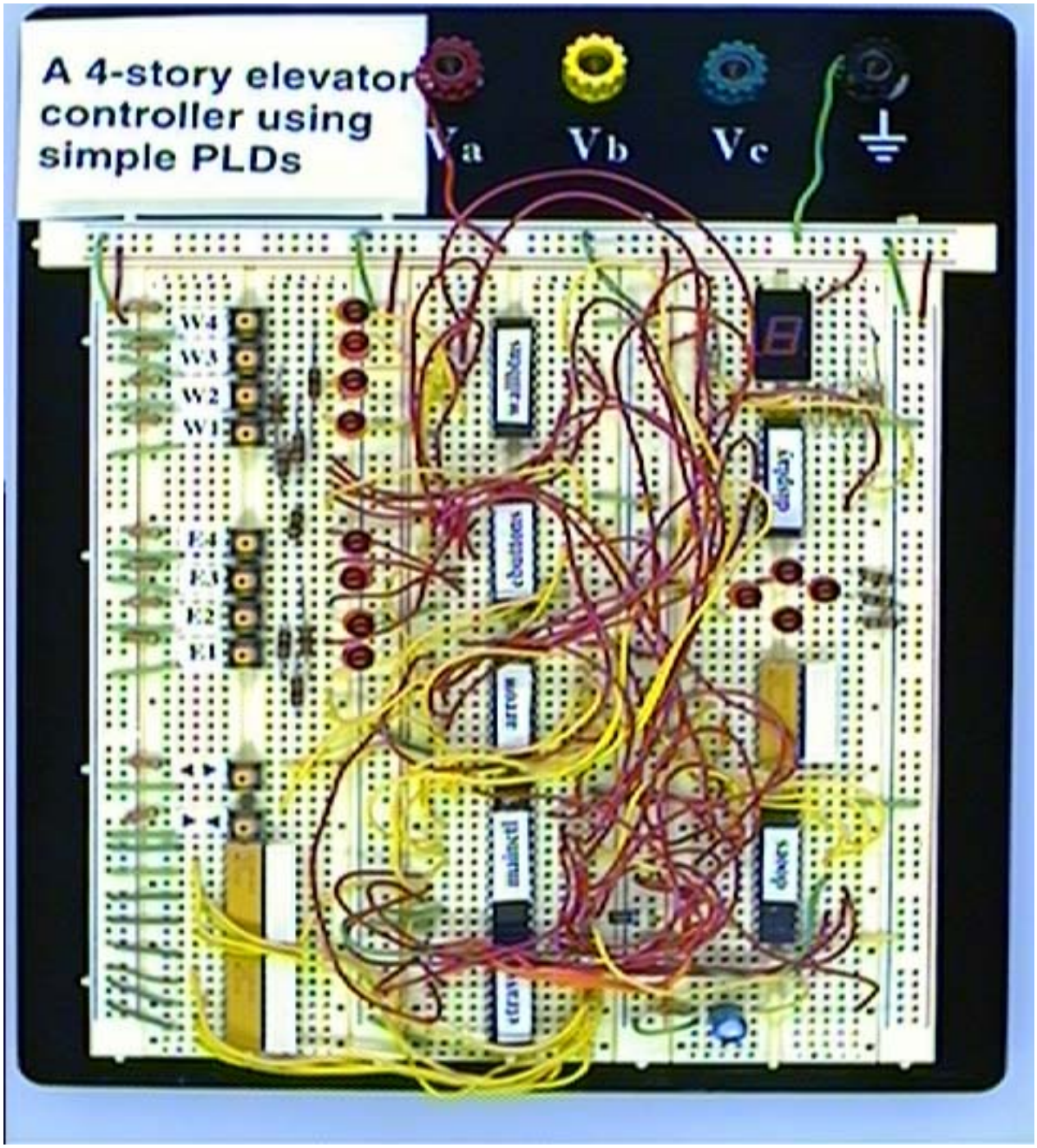

Fig. 2 The completed elevator controller and user interface circuits.

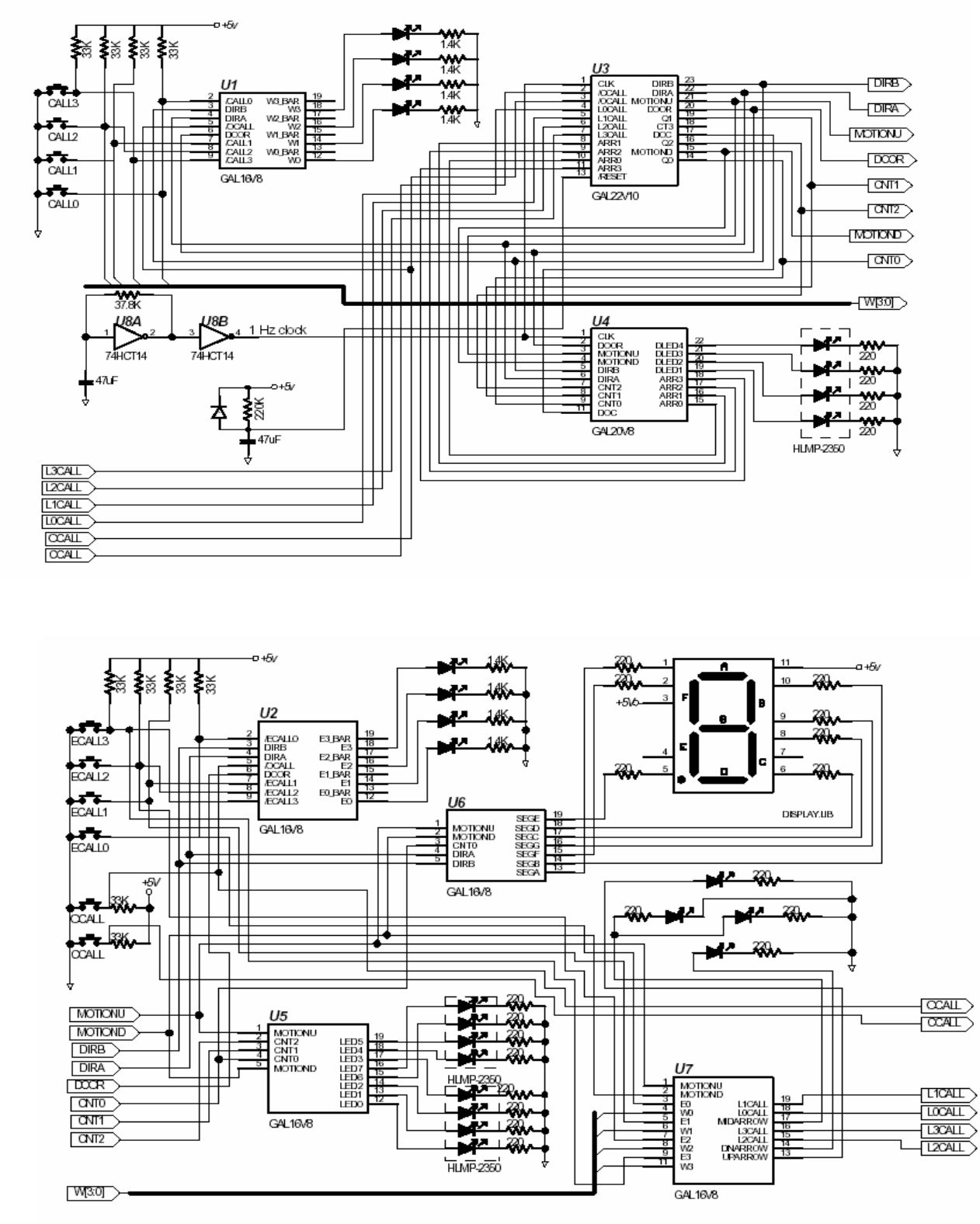

Fig.3 Schematics of the elevator controller and user interface.

The merged program (elevator.abl) retains the original design methodology. By using a synthesis tool such as XILINX ISE, the program can be synthesized and fitted to an XILINX XC9572 CPLD. The I/O signals are automatically assigned by the ISE tool. The completed circuit using a single XC9572 CPLD is shown in Fig. 4. All the above ABEL programs can be downloaded form the author's website at www.egr.up.edu/contribu/lu.

Conclusion

This simplified elevator controller design example not only can provide students an interesting hands-on state machine design experience, but also can provide them the hands-on experience in implementing a design using different device architectures. After power-up, it will be waiting at the first floor for service calls. In responding to a request call, it will open and then close the doors, travel to the target floor, flash the next arriving floor, and stop at the desired floor, just like a real-world elevator will do. It is quite a satisfaction and sense of accomplishment watching the autonomous elevator system operates exactly as designed.

This 4-story design example can be extended to 8- or 16-story elevator controller or multiple-car elevator controller. Students not only can learn state machine design, but also complicated digital system design ability through such an interesting and challenging hands-on design experience.

References

1. John F. Wakerly, "Digital Design Principles and Practices", Third Edition Updated, Prentice-Hall, 2001.

2. M. Morris Mano, "Digital Design", Third Edition, Prentice-Hall, 2002.

3. Michael D. Ciletti, "Advanced Digital Design with the Verilog HDL", Prentice-Hall, 2003.

4. XAPP328, "Design of an MP3 Portable Player Using a CoolRunner CPLD", Xilinx, 2000.

Biographical Information

Wayne Lu received the Ph.D. degree in Electrical Engineering form the University of Oklahoma in 1989. He has been with the University of Portland since 1988 and currently is an associate professor of Electrical Engineering. Dr. Lu's primary research interests are ASIC design & prototyping, real-time image processing, and dynamic scene analysis.

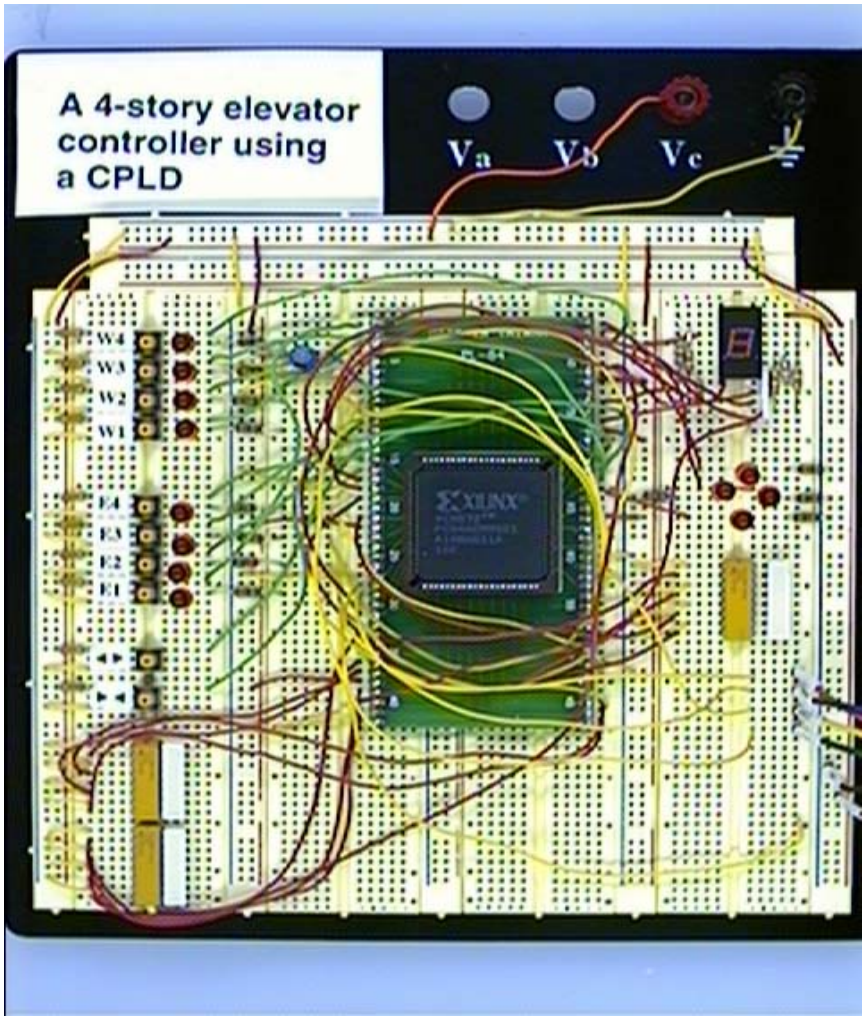

Fig. 4 The elevator system implemented using a CPLD.