Int. J. Engng Ed. Vol. 14, No. 4, p. 300–304, 1998
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Laboratory Design Projects for Secondary School Students*

BILL MONAGHAN
Department of Applied Sciences, College of Staten Island, City University of New York, Staten Island, NY 10314, USA. E-mail: monaghan@postbox.csi.cuny.edu

There are many ways to disseminate the principles and practices of engineering to prospective college students. One approach is to develop programs that actively engage secondary school students in a laboratory design experience. This paper reports on such an effort. The design and implementation of a microprocessor-based chip tester is shown to be an effective way to introduce these students to engineering. The hardware interface, the testbed, is designed and tested manually. It is then connected to the system bus of a single-board 8086-based computer. The testbed easily accommodates various chips for the unit under test (UUT). The driving software uses an exhaustive strategy. A suite of test vectors exercises the UUT for all input combinations. A table lookup compares the actual responses to the correct ones. The UUT receives either a PASS or FAIL designation. A 7400 series chip tester and a universal shift register tester are realized. Participants' reactions to the experience are reported and used as a measure of the project's success. Future extensions and a more formal assessment procedure are presented.

OVERVIEW

FOR SEVERAL summers the Science Discovery Center at the College of Staten Island, City University of New York, has been supporting a program to make the engineering experience available to schools. This program is sponsored by the New York State Education Department under a Dwight D. Eisenhower Title II Project; it also receives funding from the National Science Foundation and the National Institute of Health. Selected high school students have the opportunity to do a project in various disciplines under the guidance of the faculty. The program meets six hours a day, four days a week for four weeks. The participants give a poster presentation to the community at large on the final day. The challenge was to identify interesting design projects for the participants.

Based upon experience gained in teaching a microcomputer SDK-86 based interfacing course, a hardware/software design project was chosen. The SDK-86 allows easy access to the system bus. The necessary lines are cabled to a solderless breadboard design station. The presence of switches, pulsers and LEDs on the design station enables one to do manual validation of the hardware interface. These features facilitate rapid prototyping of the hardware interface. All input and output devices are treated as IO ports. The SDK-86 has on-board serial communication capability with an RS-232C port. Software can be developed on a PC and downloaded to the SDK computer. This feature allows rapid prototyping of the software component of the project.

* Accepted 2 March 1998.

The question still remained: 'What is a viable design project?'.

Boolean algebra—7400 series ics

Since the Boolean AND, OR and NAND functions were known to all the participants, it was natural to introduce their hardware counterparts: the quad two-input 7408 AND, 7432 OR, and 7400 NAND chips. The design station facilitates the manual testing of each chip for 'goodness'. A test vector suite exhaustively exercises the unit under test (UUT). The response vector is compared to the 'true' response vector, resulting in either a PASS/FAIL designation for the UUT. Since the above three chips are pin-compatible, each can be tested by simply using the appropriate true response vector.

SAGE: smart automated gate evaluator

The use of Automated Testing Equipment (ATE) is pervasive within the integrated circuit chip industry. The goal of the project is to automate the testing procedure; that is, design and implement a multichip chip tester for the three chips. SAGE, the realization of the above, has been reported in the literature [1] and will be briefly summarized.

SAGE: hardware considerations

A major component of the interface is the decoder for port addresses. A 74154 decoder is used in the implementation. This decoder has two enable pins, E1* and E2*, and will decode four address lines. Its wiring is shown in Fig. 1. The output port addresses are asserted low. For the decoder to be enabled, the M/IO* input must be low. The port addresses are therefore in 'IO space'.

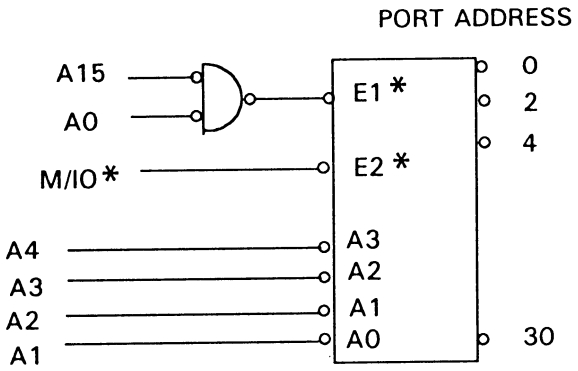

Fig. 1. 74154 decoder.

The M/IO* signal is low when the software is executing either an IN or OUT instruction.

Simultaneously, both A15 and A0 must be low. The SDK-86 board has several IO ports. The addresses for all on-board ports have A15 high. A low in the SAGE design will avoid any possible conflicts with these ports. The reason for A0 being low is more subtle. If A0 is low, all decoded addresses will be even. The 8086 processor is designed to sample the low portion of the data bus, D0 to D7, for the transfer of 8 bits of information when the port address is even.

One final comment is in order. The design for port address decoding is non-absolute. Many possible port addresses will appear equivalent to the decoder. The decision to accept this outcome was predicated on the need for simplicity within the existing time constraints. The ramifications of non-absolute decoding were thoroughly discussed and compared to a full absolute decoding implementation.

In the SAGE implementation, the port address 0 is chosen for both input and output. Two 7475 level-sensitive latches are used to capture the test vector. A 74125 tri-state 'electronic switch' isolates the results from the data bus. The UUT can be any of the pin-compatible chips. The completed SAGE hardware is given in Fig. 2.

SAGE: software considerations

The software is relatively straightforward. All chips are exercised with the same test vector, 11100100. A stored table of correct responses is constructed and used to compare the outputs of the UUT. Each table entry is one byte wide (8 bits) and contains the appropriate correct responses for the chip. For illustrative purposes consider the OR entry in the table: 01101110. The rightmost four bits, low nibble, is the correct response to 11100100. Three 2-bit right circular rotations of the test vector will generate all the remaining vectors needed for an exhaustive testing of a given UUT. Three 1-bit right circular rotations of the appropriate correct response vector will move the correct responses into the low nibble.

The IN and OUT instructions are the crucial components of the software. These instructions

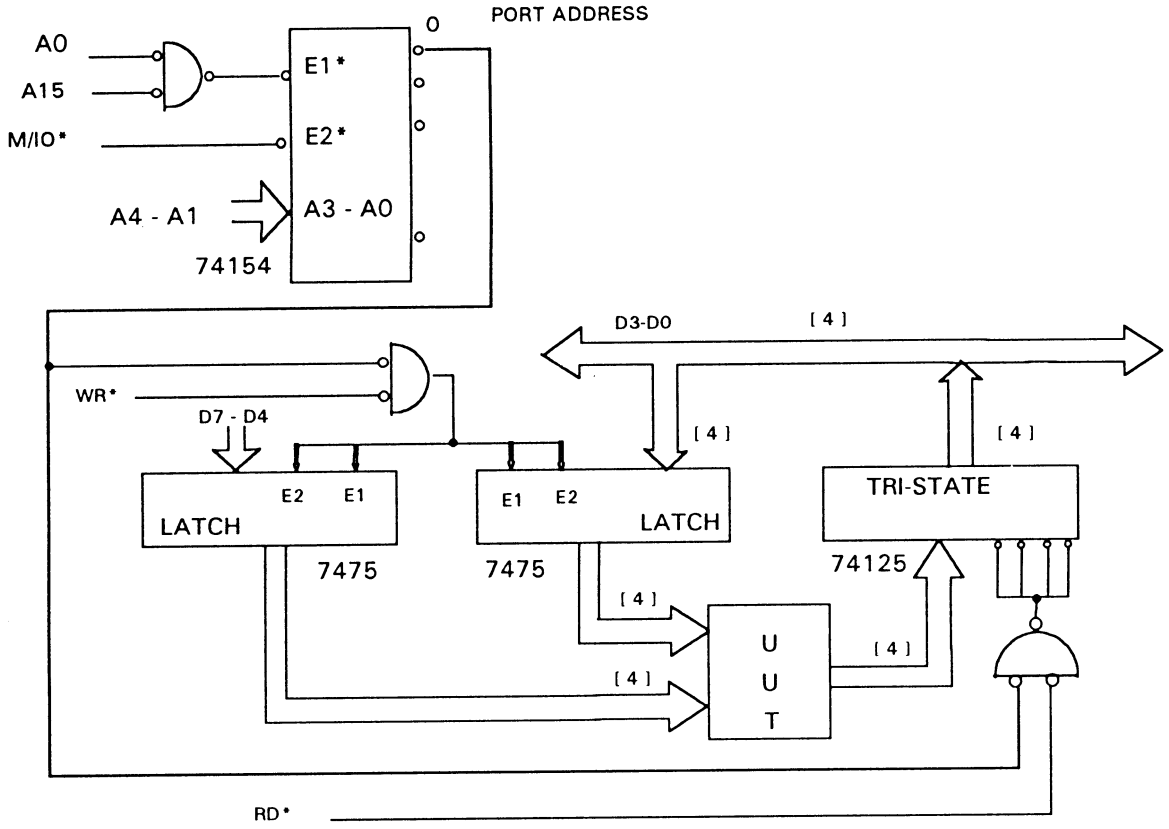

Fig. 2. SAGE realization.

require that the address of an input or output port be placed in the DX register of the microprocessor prior to their execution. A typical sequence for an input is:

```
MOV   DX, 0
IN    AL, DX
```

The DX register is 16 bits wide while AL is 8 bits. The semantics of IN can be stated as follows:

1. The content of DX is placed on the address lines A15 . . . A0 and sent to the system bus.
2. A control signal M/IO* is asserted low and sent to the system bus.
3. The read control signal RD* is asserted low and sent to the system bus.
4. The microprocessor waits for signal stabilization and then samples the data bus.

Since the outputs of the tri-state device in Fig. 2 are connected to the data bus, the above sequence will read these outputs into the AL register. Thus the outputs of the UUT are read and compared to the correct response vector.

The output sequence is quite similar to the input. Steps 1 and 2 are unchanged. Steps 3 and 4 become:

3. The microprocessor places the content of the AL register on the data bus.
4. After all signals stabilize, the processor issues the WR* signal.

This procedure will load the external 7475 latches with the contents of AL and this test vector is used as input to the UUT.

USReT: universal shift register tester

A more challenging project is available for those participants who successfully complete SAGE. USReT calls for the design and implementation of a Universal Shift Register Tester. The 74194 is a 4-bit bidirectional Universal Shift Register (USR). Two of them can easily be cascaded to form a 8-bit register. The chip has an asynchronous master reset input, MR*. The operating mode of the register is determined by two inputs, S1 and S0, and is given by the following table:

S1	S0	Operating Mode
0	0	Hold
0	1	Shift right
1	0	Shift left
1	1	Parallel load

A clock pulse input (active rising edge) performs the chosen operation. DSR and DSL inputs are used in shift operations. The parallel load inputs are designated Q0 to Q8, Q0 being the leftmost bit. Since USReT is realized after successful completion of SAGE, the SAGE interface is modified as shown in Fig. 3. Particular attention should be given to the various port addresses. Output port address 2 is used to perform a master reset of the USR and output address 4 clocks the USR.

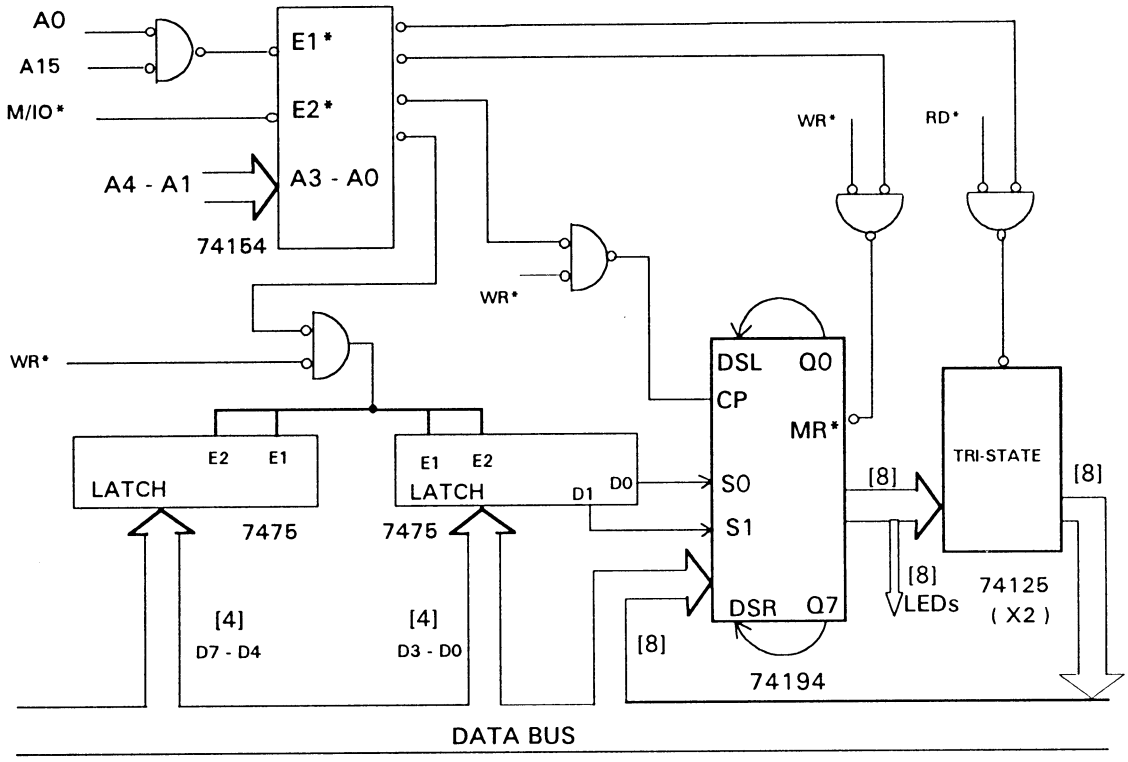

Fig. 3. USReT/ATE interface.

Output address 6 controls the 7475 latches and input port address 0 reads the USR.

USReT/ATE interface

The current design utilizes the 7475 latch as a mode register. The D_1D_0 bits written to this register determine the operating mode of the USR. As an example, the following instructions will place the USR in parallel load mode:

```
MOV   AL, 3
MOV   DX, 6
OUT   DX, AL
```

If the following instructions are now executed, the USR will latch the 2EH datum:

```
MOV   AL, 2EH
MOV   DX, 4
OUT   DX, AL
```

At any time the latched contents of the USR may be read by executing:

```
MOV   DX, 0
IN    AL, DX
```

In a similar fashion a master reset of the USR is accomplished by executing:

```
MOV   DX, 2
OUT   DX, AL
```

A HOLD operation is achieved by writing a 0 to the mode register. The sequence of *a read, a write* followed by *another read of the USR* will not show any change in its stored value. Writing a 1 or a 2 to the mode register will configure the USR for shift operations. DSR and DSL are typically independent inputs to the USR; tying Q0 to DSL and Q7 to DSR produces circular shift operations. This is a useful arrangement for testing purposes. Similar to the SAGE interface, this hardware can also be developed and tested in manual mode.

USReT software considerations: parallel load testing exhaustive technique

A 3H is written to the mode register. A loop is then executed that performs *a write to the USR*, *a read* and then *a compare* between a copy of the written value and the read value for values between 0 and 0FFH inclusive. Any discrepancy indicates a faulty USR. By introducing timing loops in the software and connecting the outputs of the USR to LEDs, the progression of the tests can be monitored. Stuck-at-1 faults are easily emulated by disconnecting the appropriate lead. This exhaustive technique loops 256 times, a strategy **not** effective for memory cell testing of millions of 8-bit locations. This realization leads into a discussion and implementation of a time-efficient algorithm for parallel load operations.

USReT software considerations: parallel load testing walking 0/1 technique

A common testing strategy is to write a field of zeroes and then walk a 1 through this field. The sequence of test values written to the USR is as follows:

```
00000000,  00000001,  00000010,  00000100
00001000,  00010000,  00100000,  01000000,  10000000.
```

Testing is performed for each value. A 0 is then walked through a field of ones. This version of testing, while not complete, is performed using 18 tests and in approximately 7% of the previous time. Both strategies are implemented in USReT; the choice of which to use is made at execution time.

USReT software considerations: data shift, hold and master reset testing

The data value, 01010101, is written to the USR. The mode is then changed to a DSR operation. Two shift operations are performed with testing after each shift. Because the wiring of the USR in the testbed produces circular rotations, a successful test indicates that each cell of the USR can effectively shift in both a 0 and a 1 from the right. The procedure is then repeated for a DSL operation.

The Hold and Master Reset testing is relatively straightforward and requires no unusual strategies. At any stage a failed test identifies a faulty USR.

Participants' reactions: initial and final

The initial overview and stated specifications for the design project generally results in nervous and fearful participants. While all have had some software experience, none of the students have had any hardware exposure. However, these concerns are resolved by using the design station to develop the testbed and exercise the basic logic chips and latches.

In the end, the participants acquire a real appreciation for the differences between input, output and control signals. The manual operation of the tester for several good and faulty chips enhances their understanding of the design specifications. It further fosters a desire to connect the testbed to the system bus and to automate the process through software development.

One reaction to the successful completion of the project is as follows: 'I produced a system which met 100% of the specifications required. It was a very eventful and exciting project to work on.' This participant had just completed his junior year in high school. He later stated that the experience was the primary motivation for his choosing to study computer engineering in college.

The parents of a second student expressed their appreciation for the opportunity the program gave their daughter to learn about engineering; she also decided to study it in college. Thus far, anecdotal evidence suggests that the program is successful in promoting engineering.

CONCLUSION

The tracking procedure currently in place shows that 92% of the participants choose a science discipline in college. Beginning in 1998, the procedure will be fine-tuned to determine how many students enter and complete an engineering program of study. In addition, departmental entry and exit questionnaires will measure the short-term objectives: how much did the participants learn about engineering and the engineering profession? Finally, feedback will be elicited on how the presentation might be improved.

Since both assembler language programming and interfacing are new to the participants, most find the SAGE project to be quite challenging. To date, however, all have completed it. The USReT project has been implemented as well, but only by the top students.

It is seen that a slight modification of the hardware testbed enables it to easily accommodate different chips. Future presentations of the program will use a 74180 parity checker and generator, 7485 comparator and the 74181 arithmetic/logic unit.

Over the past several years, our experiences with the SAGE and USReT testers clearly show that microcomputer-based hardware/software design projects are viable for secondary school students. Furthermore, engineering concepts take on real meaning in their lives, perhaps adding to the pool of future engineers.

REFERENCE

1. W. Monaghan, Smart Automated Gate Evaluator, *Proc. Middle Atlantic Section Meeting,* ASEE, 1994.

Bill Monaghan is an associate professor in the Department of Applied Sciences, College of Staten Island, of the City University of New York. He develops and teaches courses in electrical and computer engineering. His major areas of interest are in microprocessor interfacing for controllers and the use of digital signal processing microprocessors. He is particularly involved in presenting the engineering profession to secondary school students.