ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE
Paper ID #36904

Motivating Students to Learn Basic Electronic Theories by Adopting Them in Different Courses

Jack li

JACK LI is an assistant professor of Electrical Engineering Technology in the School of Polytechnic at Purdue University Fort Wayne. He earned his BS, MS, and PhD degrees in electronics engineering. Dr. Li may be reached at lij@pfw.edu.

Motivating Students to Learn Basic Electronic Theories by Adopting Them in Different Course

Jack Li, Purdue University Fort Wayne

Motivating Students to Learn Basic Electronic Theories by Adopting Them in Different Courses

Introduction

Interest in a field is a key factor to motivate students in studying [1]. Although most new technologies used in our daily life raise students' interests in studying in Electrical Engineering and Electrical Engineering Technology (EET), a lot of students lost interests in studying in this field continuously after about two years on learning basic theories. There are a lot of reasons according to student survey. A common complaint is that it is hard to find real application examples of basic theories in textbook in their life frequently. For example, bipolar transistor circuits are widely replaced by Integrated Circuits (IC); Field-programmable Gate Arrays (FPGAs) replace 7400 series Transistor-Transistor Logic (TTL) chips in most market. Even digital circuit design methods are switching to using computer languages, such as VHDL rather than basic logic design methods covered in textbooks. Another reason the author found in most classes is that some students forgot the materials they have learned after they finished the course. Most basic theories, such as Ohm's law are mainly introduced as math equations with more math calculation in one course, and are seldom reviewed or used in following courses. To most students, one course just covers several theories, and all these theories have less relationship to others. When they study junior and senior level courses, they do not know when and how to use basic theories in following courses and real-life applications. That is why most important basic theories are treated as useless materials by some students. All of these lead them to make lower performance in their major courses and it is difficult for students to face their real work quickly after they graduate from college. This paper proposed an example of adopting basic theories in different courses especially in junior and senior level courses frequently; in which digital circuit design method using a truth table is adopts to help EET students design software for a microprocessor system. Because basic theories are frequently used, students have a deep understanding of basic theories, too.

The feedbacks from students are positive. They found that the old basic theories as they called were still useful in modern technology. The real applications of basic theories in following courses motivate them to learn these "tedious" theories as well as helping them to refresh their memory to build a solid foundation of knowledge in their college education.

Problems in software design for a microprocessor system

Introduction to Microprocessors or Embedded Systems becomes an important core course of EET program because embedded systems are widely used in our daily lives, from a TV remote controller to a smartphone, and even a self-driving car. A microprocessor or embedded system is a computer system which combines hardware and software together. Besides basic computing functions, the software should co-operate with hardware, which makes program in an embedded system different with most application programs of a personal computer (PC). In an embedded system, most functions can be described as a multiple input and multiple output digital system.

The output signal depends on several input signals and even some output signals. It is does not matter how to implement the function, using hardware and/or software, the function shows parallel operation. In other words, it does not matter input signals change or not, the function should be implemented using all input signals at the same time. It is different with function description in human language, in which all conditions (input signals) are described separately in a certain sequence. For example, a thermostat in a heating, ventilation, and air conditioning (HVAC) controls the system according to real temperature inside of building and set point (or desired temperature) as well as system's operation mode, whereas the explanation of set point and system's operation mode are normally described at different section as shown in product operation manual. It is does not matter how the function is described, any function of an embedded system should be implemented according to all possible combinations of input signals and event previous output signals at the same time. Although most input information can be represented by logic functions in which there are only two different statuses for every input signal, all logic combinations of input signal might be too complex to be accounted for in design process by an engineer sometimes. That is why there are a lot of bugs in hardware and software design. It is a big challenge to most students especially for EET students because this is their first time to put hardware and software together.

Normally students in an Introduction to Microprocessors class should have learned basic C programming and digital circuits. When they design C program to control a microprocessor, "*if … elseif …*" statement was almost the only one statement used in students' program rather than using other efficient ones. It may be because "*if … elseif …*" statement is closer to daily language. Another reason is that students forgot most materials in C program. According to students' feedback, they just learned all basic C instructions and statements in limited class time without more practice related to real problem, such as control hardware using a microprocessor.

In order to help students to write more efficient code and to keep their interests in EET filed, basic digital circuit design methods [2] are adopted in Introduction to Microprocessors class to help students to learn how to use basic theories in previous courses, such as a truth table to describe real problem, and to write program in different ways as well as trouble shooting their program. All these practices help students review basic theories and use them in problem-solving, which help students realize that they really have learned useful information in college education.

Truth table in programing embedded systems

Here is one example to show how to use basic theories of truth table in digital circuit design to help students in program design and software debugging in an LCD backlight control process. A battery powered product needs an LCD display, such as a laptop. In order to save power, one function of the software is to control the backlight of LCD, which consumes more power when the backlight is on. There is a switch used to turn on or turn off the backlight manually. The backlight is also controlled by the status of battery. The backlight should be turned off and a LED should be turned on as a warning signal when battery is low.

This is a two-input (a backlight manual control switch and a battery status signal) and two-output (an ON/OFF signal to turn on/off the backlight and a warning signal to control the warning LED on/off) system. It is not hard to implement the backlight ON/OFF signal, which is controlled by manual switch signal and battery status signal. But the warning signal is not described clearly. The easy way is to turn on the warning LED whenever battery is low. If this is true, why is the battery low signal not used to control warning LED directly? This problem is hard to be found just from above system description because it only describes when the warning LED is turned on. There is no description about the condition to turn off the warning LED. This is the reason that the code may not work properly if "*if … elseif …*" statement is used to simulate human language description. It is easy to miss some conditions. If a truth table is used to describe the function, it will definitely help to raise this question before program design.

In class, students are asked to have a quick review of truth table, and then the instructor help them use a truth table to describe above problem as described in following example. In order to express easily, the following definitions are used: 1) Manual switch signal is called SW, and SW = 1 when the switch is closed which means to turn on the backlight using manual switch; 2) Battery status signal is called BT, and BT = 0 when battery is low; 3) Backlight ON/OFF signal is called LIT, and the backlight is turned on when LIT = 1; 4) warning signal is called WN and the warning LED will be turn on when WN = 1. Signal definition depends on real hardware signal. Normally 1 represents a high voltage while 0 means a low voltage. According to signal definition and system definition, a possible truth table is given as Table 1 for this example.

Table 1 Truth table of an LCD controller (1)

SW	BT	LIT	WN
0	0	0	?
0	1	0	0
1	0	0	1
1	1	1	0

LIT is easy to figure out: a) LIT = 1, which means turning on the backlight when battery is high (not low) and manual switch wants to turn on the backlight (e.g., SW = 1, BT = 1). It is not necessary to turn the warning LED (WN = 0); b) The backlight should be off when battery is low even manual switch tries to turn it on (e.g., SW = 1, BT = 0), and a warning signal, WN = 1 should be given according to system definition; c) It does not matter the battery is low or not, LIT = 0, which means turning off the backlight when manual switch SW = 0 because SW = 1 means turning on the backlight (e.g., SW = 0); d) Even battery is not low (BT = 1), LIT should be 0 when backlight is turned off by the manual switch (e.g., SW = 0, BT = 1) and no warning LED should be on (WN = 0).

It is easy to find that there are only three possible conditions for warning signal rather than four conditions according to the truth table method. The system does not clearly give warning signal's

definition of the following condition: when manual switch turns off the backlight and battery is low (e.g. SW = 0, BT = 0), what should be the warning signal? It is marked as "?" in Table 1. The warning signal might be 1 to give warning light because the battery is low (BT = 0), or can be 0 (e.g. no warning light is turned on) because the backlight is turned off (SW = 0) and there is no large power consumption. Both answers are reasonable. The customer of this project asked WN = 0 because the customer thought that it was easy to notice the warning light at the moment when people tried to turn on manual switch.

It is a common problem to miss some conditions when system description using human being language because it is assumed that all undescribed problem can be figured out using common sense of human being. This is true most time. For example, system description says, "The backlight should be turned off and turn a LED on as a warning signal when battery is low", which implies that backlight can be turned on if battery is not low. But common sense is not always correct, such as there are two reasonable choices in this example. Which one is the customer asks for? It should confirm with the customer before program design, especially when there are some definition not given clearly.

Using truth table can reduce this kind of risk because a truth table list all possible conditions by all possible combinations. It does not matter some combinations have real meaning or not in the problem, a truth table posts the possible condition. It is better to be concerned than to be ignored. Any ignorance of the possible condition may lead bugs in the system in future. Although some bugs can be solved through upgrading software, the better solution is to discover them and to solve them as early as possible.

Table 2 Truth table of an LCD controller (2)

SW	BT	LIT	WN
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

A new truth table according to the customer's requirement is shown as Table 2.

Although a truth table is widely used in digital circuit design, it can help in system description as discussed in above problem. Through this practice, students realize that the basic theories are not only used in those introduction courses, but also a method to solve the similar problems and foundation of other high level courses. These possible applications are seldom mentioned in those introduction courses. Students also use truth table in their program design in different ways with the instructor's help. C program is used in following part in order to explain the idea.

It is easy to write program according to the truth table rather than just following system description because the truth table already lists all possible combinations (cases). Most students can realize that a quick way to design program for above LCD control problem is using *switch…*

case statement to implement a state machine if a truth table is available. It is similar as a word-to-word translation. For example, if signal SW and BT are used to form a binary number named *var1*. Table 2 can be translated into following *switch … case* statement.

```
switch (var1) {
        case 0: LIT=0;
                WN=0;
                break;
        case 1: LIT=0;
                WN=0;
                break;
        case 2: LIT=0;
                WN=1;
                break;
        default:LIT=1;
                WN=0;
        }
```

Instructor also help students to read Table 2 in different ways which can help student to implement the truth table using different C statements, such as combination of *switch…case* statement and *if…else* statement as described in following two options. It helps students review most C statements with more practice, and also increases students' interests to find more and better solutions to real problems.

A. Option 1:

```
switch (SW) {
        case 0: LIT=0;
                WN=0;
                break;
        default: LIT=BT;
                WN=!BT;
                break;
}
```

B. Option 2:

```
if (SW) {
        LIT=BT;
        WN=!BT;
        }
    else {
        LIT=0;
        WN=0;
}
```

The program implementation can be a different way to re-write the truth table as shown in above two possible option codes. The program should be an error-free code because all possible conditions are implemented. This may be more efficient for multiple input and multiple output systems, which is common in an embedded system.

Furthermore, because the truth table is also used to simplify the logic equation/expression of any output signal in digital circuit design, such as using Karnaugh maps (K-map) which is a different format of a truth table or Boolean algebra to finish logic simplification [1]. If a system function can be expressed using its truth table, the system function can also be simplified in the same way as used in digital circuit design. For example, the logic function in Table 2 can be simplified as: *LIT = SW AND BT*, while *WN = SW AND (NOT) BT*. *AND* and *NOT* are logic operation in digital systems. According to this, the program can be implemented by two assignment statements as following:

```
LIT = SW && BT;
WN = SW && (!BT);
```

All above codes implement the required function. The truth table will have more combinations if there are several input signals are used. For example, there are 16 combinations for a four-input system, and the *switch…case* statement will have a long code and may not more efficiently in code. Option 1 or 2 might make code short because one input signal is used to control program sequence. Using simplified code might make code using less memory. In other words, using truth table in embedded system programing gives program designer more choices to write an error-free program.

Students are encouraged to compare different codes using above four methods by testing codes using a Renesas SK-S7G2 microcontroller board. GNU toolchains 7.2.1_2017q4 in Renesas Synergy development package V1.7.8 was used to build binary machine code which is loaded to a SK-S7G2 board to verify system functions. All programs are compiled under DOS commands in order to make the program simple. The compile commands and the start code are described in [2].

Figure 1 shows the basic code used to test. Pushbutton S4 on the SK-S7G2 board is used to simulate SW signal and S5 for BT signal. Green LED connected to P600 is to simulate backlight ON/OFF signal while Red LED connect to P601 is used the Warning LED [3]~[6]. The code just shows one quick way to convert hardware signal into a 2-bit variable *var1* shown in above code. It also converts two 1-bit signals, WN and LIT into output signal for Port 6 to control the green and red LED. The codes related for different method are inserted into the while (1) loop as commented position. The program is compiled using the same method as described in [3]. All four methods gave the same result, but the executable machine codes have different size as shown in Table 3.

```
/* Push button S4 and S5 are used to simulate SW and BT, respectively.  */
#define P0CNTR2 (*(volatile unsigned long*) 0x40040004)
/*
** Green LED connected to P600 is used as back light ON/OFF signal,
** while Red LED connected to P601 is used as Warning LED signal.
** The control signals for LEDs are opposite of paper definition.
*/
#define P6CNTR1 (*(volatile unsigned long*) 0x400400c0)

/* Mask used to keep b6 (S4) and b5 (S5) according to SK-S7G2 schematic */
#define Switch4Mask          0x00000040
#define Switch5Mask          0x00000020

void Reset_Handler(void) {

    unsigned long var1, SW, BT, LIT, WN;

        while (1) {
        /* Read port 0 data into variable */
        var1 = P0CNTR2;
        /* Adjust variable to get SW and BT signal */
            SW = (var1 & Switch4Mask) >> 6;
            BT = (var1 & Switch5Mask) >> 5;
        var1 = (SW << 1) + BT;
        /* Copy and paste the code for different method here */

        /* End of the code for different method */
        P6CNTR1 = ((((!WN) << 1) + (!LIT) ) << 16 ) | 0xFFFF;
    }
}
```

Figure 1 Test code for using a SK-S7G2 Evaluation board

Table 3 just shows that different method results in different program size, therefore, in different speed of the program was well as in different memory size. Although there is a little difference in program size, students are excited to see this difference. It motivates students to try their best to find a better solution.

Table 3 Machine code size in byte for different methods

Switch…case	Option 1	Option 2	Simplified Code
208	204	204	196

Software debugging is a very important step before the final code is released. It can find and solve the possible bugs in program. The system is easily tested just following the system description. For example, there are two function descriptions for the backlight ON/OFF: 1) the backlight can be turned off by manual switch; 2) the backlight can be turned off when battery is low as described at the beginning. Most functions can be tested following the above description. The possible combinational conditions of 1) & 2) in Table 2 might missed. The common

problem is to test function 1) when battery is low or high depending on what kind of battery status is when the system is under test. The correct way is to test the function 1) when battery is low and re-test it when battery is high. In other words, the system should be tested/debugged under all possible conditions independently. There are many combination conditions for a multiple-input system, which may or may not listed completely in system description as we described early. Some of them might be missed, especially some unreasonable combinations. All possible combinations, reasonable or unreasonable ones maybe happen in real life because of wrong operation and noises in circuits. Using the truth table to check program function should be a robust test because of following reasons: 1) A truth table lists all possible conditions of a system. It forced test process to be finished under all possible conditions; 2) A truth table list the possible conditions in a sequence, such as an increasing binary number that without any relationship with the real operation, which makes testing process following a different logic as design process does. In other words, all possible tests are independent on others. Because of this, the possible bugs in design process may be found in test process. 3) The test method can be finished automatically and quickly. For example, a counter can be used to create the input signal combinations in binary format to perform automatic test.

Conclusion

This paper describes using basic theories in digital circuit design in program design for an embedded system, in which there are a lot of hardware and/or software combination problems. It helps students to consider all possible problems before and during software design process. The truth table raises all possible conditions at the same time, and it gives students more options to implement the function, too. It makes software design with less bugs as well as reducing software debug time. It gives students a different view of software design, too. The method also helps firmware designers, who may have more circuit design background with less program experience to finish software design with less error quickly.

Other basic theories are also reviewed in this class. For example, students learned to read data sheet for component and to find proper parameters when they use Ohm's law to design LED display circuits. Fan-in and Fan-out conception are used to design interface circuit to a microprocessor system. Adopting basic theories in different courses, especially in junior and senior level courses helps students review material they have learned and gives them a deeper understanding of basic theories, which help them to use these basic theories in real problem solving. It helps keep students interested in basic theories learning as well as motivating them to keep their interests in EET program. As a student wrote in student survey, "Introduction to Microprocessors class helps me review most materials I have learned, it gives me a deep understanding of those tedious math equation. Yes, they are really useful information rather than math problems when I learned them". According to student feedback, the following suggestions are encouraged to use: 1) A quick review of the basic theories is given before using them. Most textbooks just mention the name of basic theories used, students have to review those materials by themselves in order to understand and use them. Most time, some students skip this review for some reasons, such as no reference book to be used. The quick review helps students to pick up

the material quickly. 2) Instructor guidance to using basic theories step-by-step. Most basic theories are introduced in math format, and most practices mainly focus on calculation to help students memorize the material. Students has less practices to use these basic theories in real application, and they do not know when and how to use them in their work. Instructor's guidance will help and train them how to use their knowledge to solve real problems.

Acknowledgement

The author would like to thank all former students in Introduction to Microprocessors classes, by whom more positive feedback were given when they used these methods in their program design. Special thanks are given to Ms. Cornelia Stegmann in Renesas for her support and to Renesas University Program for providing the SK-S7G2 Starter Kit.

References

[1] M. Ainley, "Connecting with Learning: Motivation, Affect and Cognition in Interest Processes." Educational psychology review 18.4 (2006): 391–405. Web.
[2] W. Kleitz, *Digital Electronics: A Practical Approach With VHDL* (9th edition). Upper Saddke River, New Jersy: Pearson, 2012
[3] J. LI, "Adopting Renesas Synergy™ S7G2 Starter Kits to Introduction of Microcontrollers Class", Renesas white paper.
[4] *Renesas Synergy Starter Kit SK-S7G2 User's Manual*, Renesas Electronics Corp.
[5] R. Oed, *Basics of the Renesas Synergy Platform*, Renesas Electronics Corp.
[6] J. Yiu, *The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors* (3rd edition). Waltham, MA: Newnes, 2014.