Teaching Systems Integration In An Advanced Microprocessor Applications Course

Phil Fabiano
New Jersey Institute of Technology
University Heights
Newark, New Jersey 07102

Abstract

The New Jersey Institute of Technology's Electrical Engineering Technology program has developed an advanced microprocessor course that teaches systems integration, and develops the skills required to minimize development design time and costs. The course uses the generic technical concepts of microprocessor technology as a vehicle to learn the basic concepts of the systems integration process. It also emphasizes the use of available and reusable technologies to develop systems and products as well as the importance of documentation and test planning. This paper describes the content of the course with regard to the basic steps of the systems integration process.

Introduction

Systems integration can be defined as the process of developing a system or product through the integration of available hardware and software technology. For example, a microprocessor based security system can be developed through the integration of an available microprocessor chip, memory chips, analog and digital devices, application programs, and I/O devices like keypads, LCD displays, alarms, and modems. The development of the system is achieved through the integration of these available technologies using relative simple analog and digital circuit design and programming.

A course in microprocessor applications offers an excellent opportunity for developing student skills in systems integration. Students that enter a senior level microprocessor course are usually required to have prerequisite courses in basic electronics, logic design, and analog circuit design. In addition, a lower level microprocessor course that emphasizes assembly language programming is required. Given these prerequisites, an advance microprocessor course can focus on the concepts, techniques and methodologies associated with systems design and integration. The steps associated with systems integration include a statement of system requirements, the top level design, the second level design, detailed design, and test planning. Each of these steps is discussed below:

Statement of Requirements

Clearly defining the overall system requirements must be emphasized as a first step in teaching the systems integration process. These requirements are often referred to as the Product Requirements Specification (PRS) and define "what" must be done as opposed to "how" it will be done. Table 1 presents an example of PRS type requirements for a motor control system.

<table>
<thead>
<tr>
<th>PRS REQUIREMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>The project must use a 12 Volt DC Motor</td>
</tr>
<tr>
<td>- Error Controlled Programmable Speed: 0 to 2000 RPM</td>
</tr>
<tr>
<td>- Percent Steady State (SS) Error: 0 % for Unit Step Input</td>
</tr>
<tr>
<td>- Percent Allowable Overshoot: 10% for Unit Step Input</td>
</tr>
<tr>
<td>- Peak Time (Tp): 0.285 Seconds</td>
</tr>
<tr>
<td>- Settling Time (Ts): 0.9 Seconds</td>
</tr>
<tr>
<td>- Output Requirements: BCD Display of Desired and Actual Motor Speed, % Error, Tp, &amp; Ts</td>
</tr>
<tr>
<td>- Concurrent Engineering: Diagnostic Test of Hardware Modular Hardware &amp; Software</td>
</tr>
</tbody>
</table>

Table 1. Example of PRS Requirements

Students tend to quickly start detailed design before having a clear understanding of all the requirements for what must be done. The tendency is to start the circuit design and coding as quickly as possible with the belief that the overall design will be completed quickly. It is quickly learns that incomplete specification of the overall requirements leads to problems that result in a significantly lengthened design cycle. Students should be taught that the start of the systems integration design process must include time for writing down and discussing with their supervisor, their understanding of the requirements for any work assignment. It is an essential element of the process to insure that what is understood as the design requirements is actually what is required by the designers customer (i.e. the work supervisor). Writing the requirements down as a formal
document that is used to guide the design should be viewed as a system requirement that is as important as the design itself.

**Top Level Design**

Given an agreed to set of system requirements, the overall design can again. As a first step in the top level design process, a basic design approach should be defined. This represents the first aspect of defining "how" the system requirements will be met. In an advanced microprocessor course this involves defining the major system hardware units starting with the microprocessor and its memory system. A microprocessor rather than a microcontroller or PC is recommended because it requires that the student deal with the details of the microprocessor bussing system and its bus timing cycles in the later detailed hardware and software design. These aspects of the microprocessor are well suited to providing the student with a design experience involving the detailed integration of hardware and software. Using a microprocessor provides the student with an opportunity for working with detailed timing diagrams in order to achieve the required hardware design. Given this experience the student will be better prepared to design digital hardware circuits involving complex timing relationships.

The top level design and its design approach should result in the following four elements of the design.

1. A first level block diagram showing the major system units and the specific technologies (eg. Intel 8086 microprocessor, "xyz" LCD display, etc.) selected for the system level design.
2. A top level flow chart showing the overall logic of the system operation and identifying, by a descriptive name the software modules and subroutines that will be used for the software design.
3. Memory and I/O maps showing the layout of the memory and the input and output devices relative to the addressing assignments.
4. A written description of the overall design approach with reference to the above three elements. As will be discussed later, it is recommended that this discussion be put on the software listing.

Figures 1 and 2 present examples of a first level block diagram and top level flowchart. Figure 3 shows an example memory map, I/O map, and programming model. Figure 4 presents an example second level block diagram.

Figure 1. Example First Level Block Diagram

Reusability is a key topic stressed as an important aspect of the top level design process. Students are taught that taking a "reinventing the wheel" design approach is not desirable. Using developed and available hardware and software wherever possible is the preferable design approach. The systems integration process is stressed as being a process that heavily capitalizes on using previously hardware and software modules. The major design job is for the most part, designing the hardware and software required to effectively "glue" available technologies and designs together to meet the given system requirements including those for costs and schedule.

**Second Level Design**

As a next step in the systems integration process, the student is taught to break the top level design down to its second level. This involves taking the first level block diagram and top level flow chart and evolving the design to a second level hardware block diagram. At this level the hardware subunits required to interface the major hardware units in the overall system are identified and defined. Examples of these subunits include address decoders and control logic for selecting and controlling the major hardware units during software program executive, initialization circuits, interrupt control and vector generation circuits, and bus interface units. The second level hardware subunits are defined in conjunction with a clear understanding of the top level flowchart and defined software modules and subroutines. Reiteration in defining the top level design and the second level design is highlighted as a natural aspect of the systems integration process. That is, the student is taught that as the design process proceeds it is quite normal that redesign occurs at each level of the design process including the definition of requirements level.
Detailed Design

Given a second level design the systems integration process can then move into the detailed design phase with detailed coding. This level should start by writing the required software modules and integrating the selected reusable subroutines. Starting with the detailed software design first is required because it is the software that determines and defines the timing diagrams associated and required for detailed hardware design.

Modularity and reusability are again stress as primary factor in doing the detailed design. The detailed design approach taught is that a main program should be written to establish required initial conditions (e.g. enabling interrupts, initialization of the stack pointer, initialization of data and interrupt vector tables, etc.). This main module functions as the overall coordinator by calling up the appropriate modules and subroutines as required.

Given a first cut at the detailed coding and resulting I/O timing relationships, the detailed hardware design can begin. This is the actual integration aspect of the system design. The student is taught that the detailed software design and its resulting timing relationships must now be integrated with the detailed hardware design. Reiteration in the design process is again highlighted as being a natural part of the process.

Completion of the detailed designs completes the paper design of the system. Students are required to document the design right on the software listing. System requirements are first written as a statement of the Statement of Requirements. This is followed by a description of the overall design approach with reference to the first and second level block diagrams, top level flowchart, and memory and I/O maps. A description of the main program that follows with appropriate references to any required figures. This is then followed by its detailed coding with line comments.

The main program is then followed by descriptions and coding for each of the modules and subroutines. Subroutine descriptions identify its calling address, user input requirements, and its output characteristics.

Figure 2. Example Top Level Flowchart
Test Planning

In order to facilitate testing the designed system, advanced test planning is emphasized. As the design process proceeds, designer awareness of testability is highlighted. The student is taught that achieving a good design includes achieving a design that is easily understood and tested. It is recommended to add additional hardware and software to achieve this worthwhile and cost saving objective. Establishing a test plan that defines the overall testing strategy, its specific tests, and the expected results are stressed as required items before starting the laboratory testing phase of the design process. The laboratory class associated with the microprocessor course supports this aspect of systems integration by requiring that pre-labs be written by the students. The pre-labs present the paper design and test plan and is required before laboratory testing activities can begin. In addition all of the code must be assembled and the hardware circuits built before testing.

Concluding Remarks

The transition in thinking broadly in systems terms rather than narrowly in a specific hardware circuit or software program design, is not an easily transition for technology students. When applying the concepts discussed in this paper, instructors should avoid basing their lectures on complex designs. Instead simple design that can serve as the vehicles to learn the basic concepts of systems integration process should be used.