APPTRONICS review

Volume 2, Issue 7, 2009 July 31

From the HoD's desk

Dear Readers,

Energy crisis is a serious concern everywhere. We have to consciously take steps towards saving energy and greening the planet. In this regard, IT industries have come up with an initiative called Green IT. Let me give some information on Green IT in this issue.

Green IT is an approach that implements solutions to achieve business and to create a healthy balance with supply /demand for IT. For this, key factors are reduction of operation cost and equipment cost, reuse and recycling. Organizations which practice such standards will achieve

•Reduced energy costs both through more efficient operations of equipment and lower usage.

•Streamlined IT processes to reduce cost inefficiencies and decrease environmental impact.

•A more mobile and agile workforce for flexible and remote working, further reducing carbon emissions from unnecessary travel.

What are the consideration for designers in Green IT ?

-Design considerations: The new designs can adopt several design practices to reduce power consumption and to reduce waste/toxic content. In electronic system design, the considerations shall start from selecting an optimal and power efficient controller and processor to the incorporation of power saving mechanisms to the design. Some of the simple design practices we can relate to are described below.

·Select energy efficient memories and storage device, use 'green IC's for power supply control chips and communication control devices.

·Use energy efficient display and LEDs, MEMS and solid state relays for industrial power electronics.

·Avoid wide external bus, and go for high speed differential serial bus.

-Process related considerations: Efficient supply chain management is the key for successful Green IT. The supply chain includes efficient process definition for deployment, tracking and retrieval of electronic equipments. There are regulations for waste disposal in most parts of the world. Mechanisms for avoiding harmful materials and costly disposal methods should be thought of. End of life management of components and products are to be outlined.

-Applications and usage considerations: It is imperative to innovate and deploy energy efficient end usage and applications. One could consider the use of thin clients with a common server at terminals where minimal processing is required . This will avoid use of power hungry CPUs at several terminals and use of energy saver utilities in your PC.

Introducing journals:

Journal of Environmental Sciences

The **Journal of Environmental Sciences** is an international journal started in 1989. It was sponsored by the Research Centre for Eco-Environmental Sciences, the Chinese Academy of Sciences.

Objectives of the Journal of Environmental Sciences

·To report on the latest research achievements and developments.

·To strengthen academic exchanges.

·To promote cooperation in science and technology.

·To contribute to the progress in environmental sciences.

The journal is devoted to publish original, peer-reviewed research papers on main aspects of environmental sciences, such as environmental chemistry, environmental biology, ecology, geosciences and environmental physics. Appropriate subjects include basic and applied research on atmospheric, terrestrial and aquatic environments, pollution control and abatement technology, conservation of natural resources, environmental health and toxicology. Announcements of international environmental science meetings and other recent information are also included.

Submission of Papers

Authors can submit manuscripts via the JES online system: ***http://www.jesc.ac.cn.*** Author's mailing and e-mail addresses should be given on the manuscript, and recommendation of two referees is appreciated. International System of Units (SI) should be used for all the data as far as possible.

It can be viewed on:

http://www.elsevier.com/wps/find/homepage.cws_home

Contents

Observations/Comments

[Let your experience teach you]

'Experience is the best teacher'. I quote this very often: during classes, lab sessions and even during friendly conversations. I used to give my interpretation for this old, but always fresh, saying also. It is not that you show your hands over a hot flame and experience the 'hotness'. It is not that you jump into deep waters without knowing how to swim to learn what follows them!

I am relating the saying with learning engineering. You come across many a failure. Do not leave them .Get the best juice out of the failures .How? Analyse :why those failures have occurred .Check for all possibilities .Through that you learn how to avoid such failures in future .This concept is not only applicable in daily life situations, but more so in your work.

Develop your attitude not to neglect even simple or minor (they are not silly) deviations or failures. That will help you to become successful .But you are confronted immediately with a question: how to develop this attitude. Well in this issue. I want to tell you one method. It has something to do with improving the lab sessions as well.

Do you get your results OK every time you do your lab experiment? No. It is good that you do not get it. There is no guarantee that if you do not face any problem or issue in the first attempt it would be so in the future attempts too. Let not the issues bubble up during a critical session like the one for the exam. If issues occur during your regular lab session it is good. In fact, you must welcome such issues. You can use them to learn. How?

You analyse the scenario. See where something could have gone wrong. List all the possible modes of failure which could lead to the specific observation you had. Check one by one. It may happen that before you complete checking all options, you get a more or less convincing mode where you get the observed 'result'. But do not stop there. Do not get satisfied. Check all options. Keep recording everything, you did and observed.

I am really happy that what I have said above is being followed, though not cent percent, at least by a few students. I am happy that this procedure is being insisted by a few of the teachers. So far so good. But things do not seem to go beyond that.

I have gone through the fair records of many. I have not come across even one case where any of those failure analyses/checks/observations is recorded. Pages in the fair records are being filled up rather in an unfair way, copying all rubbish from some purposeless laboratory guides (manuals!). This situation must change.

Continued on page 4.....

Watching Science Fiction Movies Could Save a Life!

How do you like the idea?A study was conducted recently to analyse the question of how science fiction movies influenced bio medical instrumentation and how the development of bio medical instrumentation influenced science fiction movies. It was done as:

History of cinema was divided into four sections based on the major events that changed the course of development of cinema. Then a relation was drawn between the development of ideas in films and the development of instrumentation. Focus was on the appearance of bio medical devices in science fiction movies and on how these films predicted instruments.

1900 to 1930 saw the introduction of sound as part of production in motion picture. A few films of this period like *The Cabinet of Dr. Caligari, and Metropolis*, relied much on biomedical science, and depicted devices well ahead of technology at that time.

During the Golden Cinema period (second historical section, 1930 to 1955) movies like Frankenstein (1930) used an electrical device to defibrillate the heart of a monster back to life. The first successful animal defibrillation was performed only at the end of 19th century and on human, not until 1947. The movie *The Bride of Frankenstein* showed a crude ECG monitor with a bar display. The time span paralleled the development of the first real ECG devices .

The period from 1955 to 1977 produced movies like The *Star Trek* in which the most famous science fiction medical devices were used, one of them being the Tricoder, which was a portable version of today's MRI scanner.

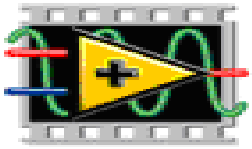
The modern period from 1977 marks the time when enormous film budgets and computing advances have allowed for spectacular special effects. In *Star Wars: The Empire Strikes Back*, there was robotic hand prosthesis installed on a movie character, to achieve nervous control and sensation. Remember: it is only a topic of investigation even now.

It has been the case with many technology advancements: science fiction predicts; technology follows.

-Sheeba Breeze

Don't bother just to be better than your contamporaries or predecessors. Try to be better than Yourself

- William Faulkner



Jeff Kodosky- *Father of LabVIEW*

For more details see the article on page 4.

ARM ARCHITECTURE

As part of our curriculum, we learn architecture and programming of eight bit microprocessor and eight bit microcontroller. 8 bit processors or controllers have 8 data lines. Each data unit consists of 8 bits. These devices have a relatively simple architecture and are good candidates to start with. But these devices are more or less outdated and now a days, most of the applications make use of 16 bit or 32 bit controllers/processors. ARM is an example of a 32 bit processor. ARM is the first RISC (Reduced Instruction set computer) processor developed for commercial purpose. Acorn developed it between 1983 and 1985. ARM stands for Acorn RISC Machine.

ARM7 is a small, 32-bit microprocessor with very low power consumption. A three-stage pipeline occupies minimal silicon area yet allows division of the execution time of each instruction into three parts: instruction fetch from memory, instruction decode, and instruction execution. The most complex stage is the instruction execution stage. A register read, a shift applied to one operand, an ALU operation and finally a register write -all execute in one clock cycle. This limits the processor's maximum clock speed to around 80 MHz on a 0.35-micron silicon process. However, that speed is more than enough for any cost-sensitive applications using ARM7.

The combined shift and ALU execution stage is an important ARM feature. A single instruction can specify one of its two source operands for shifting or rotation before it is passed to the ALU. This allows very efficient bit manipulation and scaling code, and virtually eliminates single shift instructions from ARM code. The ARM processor does not have explicit shift instructions; a move instruction applies a shift to its operand if needed.

ARM7 also uses a vonNeumann memory architecture; the instructions and data occupy a single address space and are accessed with individual address and data buses. Though this limits performance—instruction fetching (and hence execution) must stop for instructions that access memory—the reduced cost of a single memory outweighs performance in many embedded applications. To reduce the penalty of data accesses stalling the pipeline, ARM implements load multiple and store multiple instructions. ARM has a load store architecture, which means all the data is accessed through registers. The instructions can move any of the ARM registers to and from memory, and update the memory address register automatically after the transfer. This not only allows one instruction to transfer many words of data, it also reduces the amount of instructions needed to transfer data. As a result, ARM code is smaller than other 32-bit instruction sets.The ARM instruction set has conditional branch instructions. These follow a test or compare instruction to control the flow of execution through the program. Conditional move instructions allow data to be conditionally transferred between registers. The ARM instruction set takes this functionality to its logical extreme, allowing all instructions to be conditionally executed. Loads, stores, procedure calls and returns, and all other operations may execute conditionally after some prior instruction sets the condition code flags. This eliminates short forward branches in ARM code, improves code density and avoids flushing the pipeline for branches, further increasing execution performance.

ARM8 is the next core in the ARM line. It extends the ARM7 implementation in two fundamental ways, with two additional pipeline stages and a new cache interface. ARM7's execute stage splits into three separate stages on ARM8, and register read moves back into the decode stage. The two additional pipeline stages perform memory accesses and register writes. Because each instruction executes over multiple cycles, register-forwarding paths must pass data between successive instructions. This is necessary because one instruction will not have written its result to the register file before the next two instructions have read their source register values. ARM8 incorporates a single cache interface that allows instruction fetches in parallel with data accesses. It retains ARM7's von Neumann cache interface, but doubles the bandwidth of the interface to provide 64 bits every cycle. ARM8 also uses a sophisticated pre fetch buffer and branch prediction unit to fetch instructions ahead of the execution unit. When the cache is not in use for a data access, two instructions are loaded into the pre fetch buffer. This allows single cache to satisfy both data and instruction accesses. Performance of ARM8 is comparable to a Harvard machine with separate instruction and data caches, yet retains the simplicity of a single cache machine. ARM8 delivers 100-MHz operation in a typical 0.35-micron process, and lowers the average number of clock ticks per instruction to around 1.5. This increases overall performance by about 70% over ARM7.

Digital Equipment Corporation code signed the StrongARM1, the fastest of our current processors. Harvard architecture delivered maximum cache throughput and a five-stage instruction pipeline allowed maximum clock rate. These features produced an embedded processor that is faster than some workstation processors. StrongARM110 incorporates two 16-Kbyte caches maintained even when the processor is coupled to a relatively low-speed memory system. StrongARM1 machines deliver 233 MHz with less than 1 Watt of power consumption when coupled with Digital's very fast 0.35-micron process, which operates with a 2- volt supply. Its power consumption/performance ratio makes it the best in the industry.

-Mary Hexy

LabVIEW (**Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**ork bench)

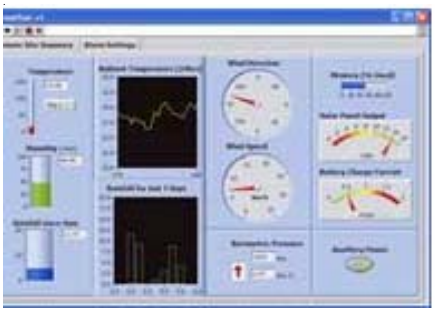
It is a platform and development environment for a visual programming language. It was developed by National Instruments. It is widely used for data acquisition, instrument control, and industrial automation on a variety of platforms like Microsoft Windows.

It is a graphical programming language. It uses icons instead of lines of text to create applications. In text-based programming languages instructions determine the program execution. But LabVIEW uses dataflow programming, where the flow of data determines the execution.

LabVIEW programs are virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. Every VI uses functions that manipulate input from the user interface or other sources and display that information or move it to other files or other computers.

The following three components are there in a VI:

• **Front panel**— In LabVIEW, we can build a user interface with a set of tools and objects. This user interface is known as the front panel. It allows the user to interact with the programme and also helps to visualize the output from the application. This panel consists of controls and indicators which allow an operator to input data into, or extract data from, a running virtual instrument.

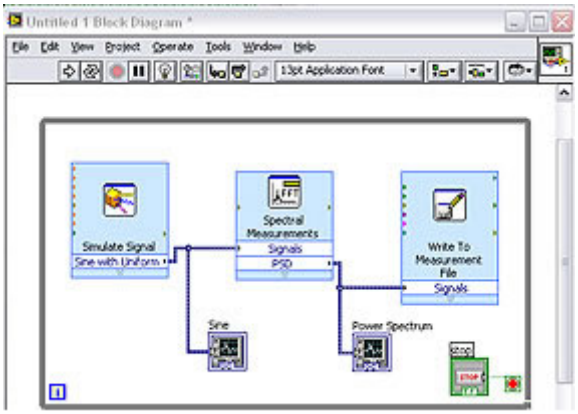


• **Block diagram**—It consists of a graphical source code that defines the functionality of the VI. Also we can add codes using graphical representations of functions to control the front panel objects. These codes also get displayed in the block diagram.

The block diagram shows how the controls and indicators are integrated. It also shows the hidden modules where all the functions get done. It makes use of graphical symbols connected through software based data lines in the form of wires.

In some ways, the block diagram resembles a flowchart.

• **Icon and connector pane**—Identifies the VI so that we can use one VI in another VI. A VI within another VI is called a subVI. A subVI corresponds to a subroutine in text-based programming languages.



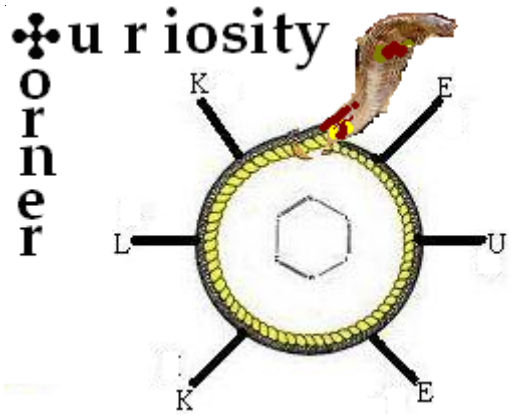
The concept in LabVIEW is "dataflow" - any item get executed when all its inputs are available. This implies parallelism (or at least pseudo-parallelism). The standard execution is left-to-right because inputs are generally on the left of an item and outpts on the right, but this is a convention, not a requirement.

The site www.ni.com gives information about various products supporting LabVIEW, applications in LabVIEW, tutorials, Demos, and Case Studies.

-Sreejith

.....Continued from page 2.

Make it a habit to keep record of all deviations/anomalies/ failures, related analyses, measures taken for correction, improvements got in results, etc. in the fair records also. It is very much required .Then only you get the true benefit of doing a laboratory exercise. In fact that must be the most important section in fair record. That helps your experience to teach you.



Discover the great connection with history of science.

Answer to the question in the previous issue

Check for Saros cycle. You will get it.