COMPANY CONFIDENTIAL

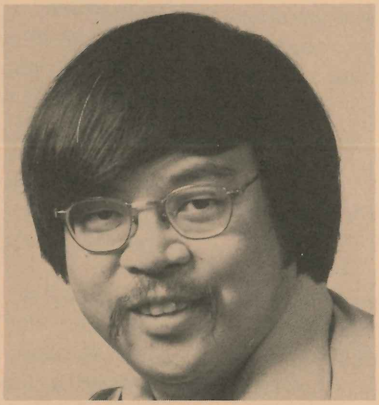
Published
15 August 1978

FORUM REPORT 8

MICROPROCESSOR DESIGN PITFALLS

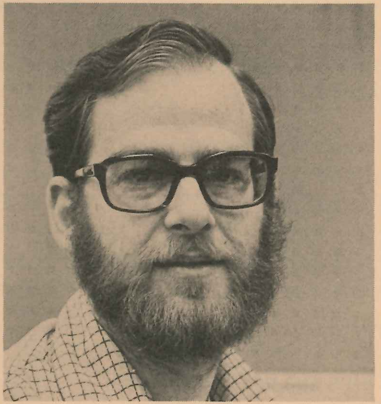
Bill Walker (T&M Group vice president), in November 1976, formed the Engineering Activities Council to provide engineers with a forum in which to present directly, to multiple levels of management, what engineers themselves consider important in technology.

The subject of the eighth forum was Microprocessor Design Pitfalls. Bill Walker introduced the forum chairmen: Paul Williams (Engineering Services, T&M Operations) and Robert Chew (Instrument Research, Tektronix Laboratories). Paul and Robert introduced the forum panel members: **Norm Kerth** (Computer Research, Tektronix Laboratories), **Bill Lowery** (Scientific Computer Center), **Steve Dum** (LDP Engineering) and **Bob Edge** (IDS Engineering) who spoke about pitfalls from an IDG designer's viewpoint.

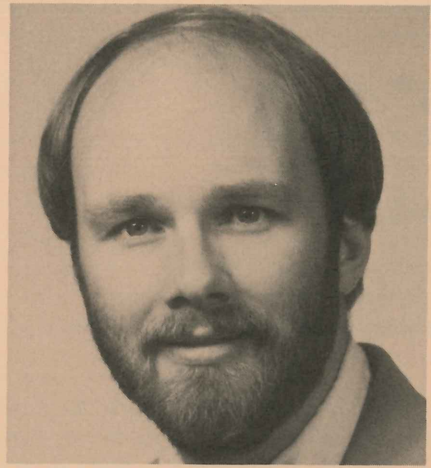

Robert Chew

Paul Williams

COCHAIRMEN


Steve Dum, LDP Engineering,spoke about logic development support for microprocessor design.

MICROPROCESSOR DESIGN PITFALLS FROM A DESIGNER'S VIEWPOINT



Norm Kerth, Computer Research (Tektronix Laboratories), ext. 6640.

From a designer's viewpoint there are several major pitfalls on the road of microprocessor-based product design. Each requires the attention of the engineers and managers involved in the project.

PRODUCT DEFINITION

In some companies, marketing people decide which products will be designed. Since most marketing people have little contact with new technology, their design suggestions tend to be warmed-over versions of existing products. The new products may be incrementally bigger, smaller, faster or cheaper but they aren't radically different in the technology they employ.

In other companies, engineers decide which products will be designed. While familiar with the latest technology, they often aren't familiar with markets. Products designed without marketing input often have little-used options and high prices.

Since the engineer's job is designing, the engineer has little time to learn the spectrum of markets that are "out there." For example, engineers know that cable testers are used to check cables in aircraft and telephone lines. But marketing people are more likely to know that a cigar manufacturer uses a cable tester to make sure each packaged box contains the right number of cigars, and that a potato farmer uses a spectrum analyzer to find rotten spots.

The engineer's job is designing, and the marketing person's job is identifying markets and selling to them. I believe that Tektronix needs for each project a third person whose full-time job is new product definition. This person should have a current engineering background, an understanding of software and an exposure to the marketplace. Job responsibilities would include complete product definition, product simulation and maintaining contacts with customers and the design groups.

SYSTEM OVERVIEW

After product definition, the next step in microprocessor-based product development should be formulating a **complete system design.** Not having a consistent, well-thought-out system design leads to several problems.

Not all design changes can or should be avoided, but having a firm system design helps engineers resist the temptation to add "just one more neat feature" that may have an adverse impact on the rest of the design.

Lack of coherent system design also sustains the "do-and-redo" syndrome. If only part of the design has been thought through, the software designer may be told to write the code for that part of the design. When other parts of the system design are completed, the software designer often has to redo the first efforts.

One way to minimize these problems is to assign a **system architect** to each project. Microprocessors have made instrument design much more complex. To meet that complexity, there should be one system architect who defines the overall design, documents that design, and answers design questions during product development. The system designer should also make sure the the product modules work together. Good system designers are rare, but we should still look for them.

HUMAN INTERFACE

Microprocessor control of the human-machine interface makes that interface very flexible, but also introduces a pitfall. It's easy to wait until the instrument is built to check out the interface. If the interface doesn't do what was expected, the designer may have to make major changes. To avoid the pitfall, the designer can simulate the interface using a 4051 graphic terminal, a bit-bucket or the Cyber system.

Fortunately, you don't have to simulate the whole instrument just the human interface. It's usually enough to mock-up the front panel... modeling keyboard action, the crt, or other human-interface devices. The mock-up will show any flaws in the human-interface design as well as allow the designer to show marketing and manufacturing people how the instrument works. And then let novices use the simulator ... that's the ultimate test.

The simulator is a useful educational tool because marketing and field people can begin learning how to operate the new product. Consider having a few customers try the simulator. After all, they are will be using and buying the product.

I know from experience that the HP 65 programmable calculator was simulated two years before engineers started the design. With the simulator, applications engineers wrote programs for the calculator and suggested changes to the keyboard even before the design engineers began their work.

PROGRAMMER PRODUCTIVITY

Any designer's productivity is a result of his own efforts and his interactions with his co-workers.

Walkthroughs can increase a designer's productivity by efficient interaction with his peers. A walkthrough is a meeting of a designer and three or four peers. In each meeting, the peers review a couple of pages of code or a block diagram of proposed code. Reviewing code, they examine the code for efficiency and the comment statements for clarity. They also look for bugs. With the block diagram, the reviewers look for alternative solutions, design flaws and design omissions.

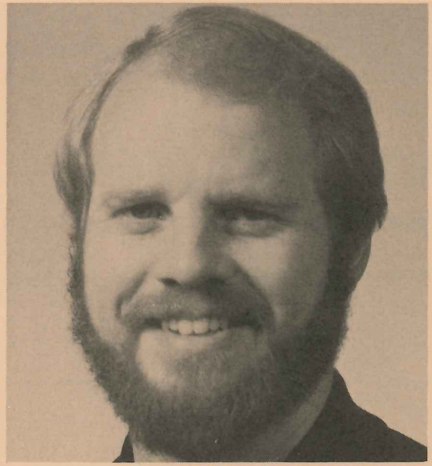
Walkthroughs have many benefits. First, more people than the designer know the program ... valuable in case the designer decides to move to Tahiti on a day's notice. Seeing other designers' work can stimulate each designer's own thinking. And walkthroughs minimize ego problems (the designer doesn't feel so bad if a bug has slipped past three other programmers too).

For walkthroughs to work well, the participants should follow a few ground rules. First: *no managers allowed*. The designer should feel that the program, not the designer, is being examined Second, *walkthrough transactions are private*. This encourages the participants to speak freely. Third, *all walkthrough participants should attend every meeting* so that they will know why decisions were made to go this way or that.

MANAGING SOFTWARE

There are a number of pitfalls for managers. Fortunately, there are a couple of good books available for managers of software designers: **The Mythical Man Month** by Fredric Brooks and **The Psychology of Computer Programming** by Gerald Wienberg. Both are available in the Tektronix library. To borrow a copy, call ext. 5388.□

PREPARING HARDWARE ENGINEERS FOR THE SOFTWARE WORLD



Bill Lowery, Scientific Computer Center (T and M Operations), ext. 5865.

To remain up-to-date professionally and to produce competitive products, hardware designers must now become familiar with the software world. The best way to enhance their software education is to provide a total learning environment - an environment that provides incentives to learning, provides learning tools, and encourages formal education.

The software education of the hardware designer serves the interests of both the designer and his employer. The employer should therefore provide as much encouragement and as many learning opportunities as possible and the engineer should take advantage of the opportunities. The designers and companies who fail to make progress in software education will surely lose their place in the market.

INCENTIVES

Even in groups of talented and accomplished designers there sometimes is inertia in designing with traditional design tools. In spite of the software revolution introduced by microprocessors, some designers may continue to use random logic for their designs simply because they lack enough incentive to learn new ways.

There are several factual arguments for hardware designers learning software techniques. **Lower costs,** for example, may be a reason for switching to software solutions to some design problems. (Costs are subject to continuous change... the costs of programming, logic and memory are all in flux).

Flexibility is another advantage of software. Sometimes, minor changes can be made to software at less cost than changing the equivalent random logic circuits. There are limits, however, to the changes that can be made economically.

Other incentives for learning software are more personal. The availability of software **training** will make most designers aware that changes are coming. **Competition** from peers and juniors who have learned software design techniques is also a major incentive for overcoming inertia, but managers should avoid producing a sour-grapes attitude in their groups (some designers may react with "I'm good at what I do, so I don't need to learn any of this new-fangled stuff").

Another major incentive for the traditional hardware engineer is the desire to become a "complete engineer" — one who is proficient with both hardware and software design. This type of designer can make valuable inputs in all phases of product development, and has a wider perspective than either the software guru or the hardware wizard.

BUILDING FAMILIARITY

Eliminating uncertainties about software makes the transistion to software design easier. Dispelling the myth that software is less exact than hardware is one approach.

Another approach to making the transistion easier is avoiding the "technology shock" that sometimes results when a designer moves from a familiar to an unfamiliar field. The former expert is a now a novice and will inevitably make mistakes at first. These first efforts should always be treated with respect.

Another barrier on the road to familiarity with software design is the "pseudo-expert syndrome" ... the belief that if you can write code in a given language, then you can write a program or even design a software system. In the hardware world, this is equivalent to saying you can design circuits if you know how to wire IC pins together. Designing a software system is as complex as designing a hardware system. The software provides flexibility, but that flexibility magnifies the possibility of making errors.

Building familiarity with software design takes time. The hardware designer has the talent, but he also needs the time to learn the fundamental concepts and to use them enough to reach proficiency.

LEARNING TOOLS

Efficient learning requires more than books. The engineer must also have hands-on experience with microprocessor design aids. Available aids range from inexpensive hardware aids, coded in machine language, to expensive system-development aids.

The most sophisticated versions of the latter are general-purpose computers with attached hardware-debugging systems. The investment required for such a system is justified for companies that are serious about developing and maintaining in-house software systems expertise.

The most sophisticated design aids provide a "friendly" learning environment which is very important for the new user of any system. It's hard enough learning software design without having to first learn a host of details for operating the aid.

ENCOURAGE FORMAL EDUCATION

It is the responsibility of serious designers to add the experience of other designers to their own experience. Formal education is one route to take in that direction.

A company can encourage formal education by paying the engineers' tuition at local schools, by sending employees to workshops and seminars, and by sponsoring educational programs within the company. Our experience in the Scientific Computer Center tells us that engineers do take advantage of in-house opportunities resulting in growth in individual and corporate expertise.

Fortunately, here at Tektronix we do have a major formal education program. Education and Training lists many software courses in its catalog, and the Scientific Computer Center offers its facilities for learning through sharing common interests. The Microprocessor Users Group is an example.☐

COMPANY CONFIDENTIAL

Managing editor: Burgess Laughlin, ext. 6792. Graphic designer: Joan Metcalf. Compiled and edited by the T&M Publicity Dept. for the benefit of the Tektronix engineering and scientific community in Beaverton, Wilsonville and Grass Valley. For information, call ext. 6792 or write to D.S. 19-313.

Forum Report is distributed over the **Engineering News** mailing list. To add your name to the list, or change your delivery station, call ext. 6792.

Maureen Key 60 553