Embedded Is the New Paradigm(s)

Wayne Wolf, Princeton University

A well-known set of applications has grown up around those capabilities: spreadsheets, word processing, and Web browsing—to name three. These are fundamentally important applications that won't go away any time soon.

The desktop computer won't disappear for the foreseeable future either. But we need to remember that as we assemble different components around the basic CPU and as we gang together powerful CPUs, the resulting system has new capabilities that we can use in new applications.

Embedded computing moved beyond toasters quite some time ago, but there are still misconceptions about what embedded computers do. Some of those misconceptions come from an old-fashioned view of what a computer is.

Embedded computing can support a world of applications not possible with mainframe or desktop interfaces.

MAINFRAME MONSTERS

The 1950s' science fiction movie *Forbidden Planet* is a prime example of one stereotypical view of computers. The movie introduced Robbie the Robot, but its main villain was a computer that was as big as the entire planet. This wasn't an embedded computer—it was the universe's biggest mainframe. The computer didn't come to the people; the people went to the computer.

We don't carry boxes of punch cards to the card reader any more, but viewing the computer as a distinct object still influences a lot of thinking about what computers are and what they can do.

Mainframes sat in a room and performed tasks like database management. The tasks involved large volumes of data, but the data was already in machine-readable form.

When powerful computers were expensive, it made sense to conserve their resources by paying people to prepare data for them.

DESKTOP DYNASTY

Desktop personal computers drastically lowered the cost of computing. Let's take a trip down memory lane.

Alto

The Alto set the form for today's desktop computer. Developed at Xerox Palo Alto Research Center in the 1970s, the Alto combined all the major components of today's PC: bit-mapped display, pointing device, local storage, and network connection.

The Alto's big insight was its attention to input and output. Most of Alto's CPU cycles went to I/O. The word processing program, for example, spent a lot of its time figuring out how to typeset the characters onto the screen and the printed page. Although the Alto didn't do things like handwriting recognition, this concept was very advanced and showed the way to a new vision of computing.

PC product category

However, it's important to keep in mind that "desktop PC" is fundamentally a product category. The computer itself is the CPU buried inside the box. The PC is a collection of components—the CPU, disk, screen, network connection, and so on—that combine to provide a specific set of capabilities.

From luggables to handhelds

Comparing two early portable computers emphasizes how system configurations and applications go hand-in-hand.

Osborne 1. Introduced in 1981, the Osborne 1 is generally credited as the first portable computer. It looked like a carry-on bag, and advertisers told us it was "so small it fits under an airline seat." (True, but what do you do with your feet?)

The Osborne was organized very much like a desktop PC. It had a 5-inch green CRT screen and two 5-1/4-inch floppy drives. It also had 120 V power cord—this machine didn't run on a battery.

You could run the same programs on your Osborne as you did on your desktop PC. That was its attraction but also its limitation. The Osborne had a relatively short lifespan. We had to wait a decade for desktop components to become sufficiently light and energy-efficient to make laptop computers popular.

TRS-80 Model 100. In 1983, Radio Shack introduced the TRS-80 Model 100, a portable computer designed for one application: text capture.

It had an LCD screen, a built-in keyboard, and a port for attaching a modem. The CPU was the Intel 80C85, a CMOS processor. CMOS was relatively rare at the time but consumed considerably less power than the standard NMOS processor. It ran on four AA batteries and came with a built-in word processing program, address book, scheduler, and Basic.

The Model 100 was wildly popular with reporters, and for several years it was among the world's best-selling computers. It was popular because it was tailored to an application and performed that job well. Reporters needed to be able to write text, edit it, and then deliver it to their editors while on the road.

The Model 100 didn't have much RAM or any disk drive. You couldn't store a book on it, but you could write an article like this one with the entire computer sitting on your lap.

I have both these machines in my personal collection of "antique" computers. By today's standards both look old-fashioned—one of my students reacted to the Osborne by saying, "You mean that's from the 80s?"

But the Model 100's small form factor and clean design come much closer to a modern device. I can imagine using the Model 100 because it fits my needs. I can't imagine holding the 25-pound Osborne on my lap.

Personal Digital Assistants. The PDA is a modern version of the Model 100. It's PC-like in some ways—Windows CE devices run spreadsheets and word processors, for example—but it is designed as a mobile device for tasks you want to do while on the move.

A PDA performs real-time handwriting recognition. It is also highly optimized to reduce energy consumption and stretch battery life.

Moreover, PDAs are designed to work in tandem with PCs. This design feature distinguished the original Palm device from previous handheld computers. You can manipulate your schedule and phone book either on the desktop or on the PDA, and you can move information seamlessly back and forth between them.

The PDA is optimized for mobile use and the more general, feature-rich user interface is reserved for the desktop.

Gesture control is a prime example of computation serving the user rather than vice versa.

CPU CAPABILITIES

But CPUs can do more than manipulate preformatted data or expand desktop capabilities. We need to go beyond the desktop to wherever the action is, whether on the road, in the sky, or on the water.

So what can we use CPUs for?

We can certainly employ them in user interface functions, such as handwriting recognition in a PDA.

We can also use them to process all sorts of streaming data. Data came into mainframes in batches, but the real world operates continuously. Today's powerful embedded CPUs can handle this constant data barrage if we're clever enough to exploit their capabilities.

We can, of course, use embedded computers to do things behind the scenes. CD and DVD players are prime examples of a trend toward using computation to correct for mechanical device limitations.

CD and DVD drives are cheap, flimsy plastic devices. Reading the data off the disk requires controlling that little piece of plastic to within micron tolerances. CD and DVD players do this by using embedded processors to execute complex control algorithms that operate continuously on the data streaming in from the read head. The typical CD player performs ridiculous amounts of computations using the power available from a couple of batteries.

Communications is another enabling technology for advanced systems. Once again, we increasingly use computation to overcome the limitations of the transmission medium. Embedded computing makes it possible to deploy sophisticated communication algorithms on battery-powered devices like cell phones.

More complex communication systems, such as those in which there are multiple antennas, will rely even more on embedded processing.

WHAT DO USERS WANT?

But let's get back to what the user wants to do with all these widgets.

As this column's title suggests, there is no single paradigm for embedded computing. Many applications use it, and each one imposes different requirements on the embedded system. Embedded computing's explosive growth only means that it has matured enough to support applications that weren't possible with mainframe or desktop interfaces.

Gesture recognition

Gesture-controlled interfaces represent one radical departure from the desktop paradigm. Our research group at Princeton has experimented with devices based on this technology, and I saw an IBM demonstration of gesture-based control a few months ago. The MIT Media Lab and many others have developed gesture-based control systems, and Sony recently announced an application for the PlayStation 2.

Gesture control is a prime example of computation serving the user rather than vice versa. Performing even modest gesture-recognition algorithms takes a lot of compute power, but abundant, low-cost embedded processors can support complicated devices to make the user's life simpler.

Inventory control

In a more prosaic vein, the communication systems that shipping and trucking companies use to track inventory provide another good example of

computation in the service of users. The underlying function is a fairly straightforward database. The problem is getting the data into and out of that database. So drivers carry battery-powered handhelds and use terminals in their trucks to track deliveries automatically.

Radio-frequency identification tags will take these systems to new levels. For example, RFID tags allow automatic readers to track inventories without requiring people to punch keys or swipe bar codes.

Scientific computing

Scientists and engineers shouldn't be chained to their computers any more than business people, nor should they be restricted to batch-mode analysis. Embedded computing can allow data processing, analysis, and visualization to track with scientific data collection.

The real-time and low-power challenges of next-generation scientific computing will require solutions that apply embedded system techniques.

Distributed processing

All these applications involve important distributed computing functionality. We generally don't solve embedded computing problems with one big CPU. Instead, we use a network of distributed processors to put the computation where it is needed.

There are several reasons to use distributed over centralized approaches—the two most potent are real-time deadlines and energy consumption.

Distributed computing complicates system design, but many applications require it. While all the traditional distributed computing methods are useful in embedded systems, we need to extend them to handle real time, limited bandwidth, and energy constraints.

Some people regard almost any information system as a desktop-style computer attached to some magic device. That view not only restricts our notion of a user interface but also ignores the design problems that all the other parts of the system pose.

The boundary between user interface and computation is increasingly blurred. Interpreting gestures, for example, requires a fairly abstract model of a person and the system state. Interpreting user needs requires more than just pixel pushing. Distributed architectures that use embedded software technology will perform all sorts of functions to meet this need. ■

Wayne Wolf *is a professor of electrical engineering at Princeton University. Contact him at wolf@princeton.edu.*