The InfoPad Multimedia Terminal: A Portable Device for Wireless Information Access

Thomas E. Truman, Trevor Pering, Roger Doering, *Member*, *IEEE*,
and Robert W. Brodersen, *Fellow*, *IEEE*

Abstract—The architecture of a device that is optimized for wireless information access and display of multimedia data is substantially different than configurations designed for portable stand-alone operation. The requirements to reduce the weight and energy consumption are the same, but the availability of the wireless link, which is needed for the information access, allows utilization of remote resources. A limiting case is when the only computation that is provided in the portable terminal supports the wireless links or the I/O interfaces, and it is this extreme position that is explored in the InfoPad terminal design. The architecture of the InfoPad terminal, therefore, can be viewed as essentially a switch which connects multimedia data sources in the supporting wired network to appropriate InfoPad output devices (e.g., video display), and connects InfoPad input devices to remote processing (e.g., speech recognizer server) in the backbone network.

Index Terms—Mobile computing, wireless communication, low-power CMOS, system integration, design, specification.

1 Introduction

There is presently a reexamination of the requirements of the system architecture and hardware needed for personal computing for the new and ever-growing class of users whose primary computing needs are to access network information and computing resources, as well as real-time interactive activities (chat rooms, games) and direct communications with other people. These applications, which are more communications-oriented than computation-oriented, require a "personal computer" that primarily has support for high bandwidth real-time communications, as well as multimedia I/O capabilities. User-accessible general purpose programmability and local high-performance computation are a secondary requirement, desirable only if the increased complexity and cost to support stand-alone operation, which is required for disconnected or poorly connected operation, can be justified.

As the dependence on network-accessible information storage and computation increases, the desire to ubiquitously access the network will require the terminal to have the portability of a paper notebook (1 lb, 8.5" x 11") while still being able to support real-time multimedia capabilities. These goals require a sophisticated wireless communications link that must provide connectivity even in the situation of large numbers of colocated users, such as in a classroom.

The InfoPad system design explores a highly optimized solution to the above goals, and critical to the design is the assumption that high bandwidth network connectivity is available (Fig. 1). The operating environment is divided into partially overlapping service areas ("picocells"), each with a coverage radius that is typically 10-30 meters, depending on the construction of the physical environment. Each picocell must provide service to approximately 50 users, and is expected to support a seamless handoff for roaming users.

These are the goals of the InfoPad system and, though not all of these specifications were met in the realization discussed here, the architecture that was developed would meet these goals with the application of even present state-of-the-art component technologies.

The primary aspect of the InfoPad system architecture that differentiates it from other portable computing systems is the role of the portable end-user device: *The InfoPad essentially functions as a remote I/O interface, instead of a computation and storage device*. Since portability and widespread consumer use was an important requirement, it was necessary to reduce the energy consumption, weight and cost of the end-user device as much as possible. For this reason, exploitation of the availability of the network connectivity and access to network servers was deeply built into the overall system architecture [2].

In our architecture, computing and storage resources are removed from the portable device and are placed on a shared, high-speed backbone network of servers that provide mass storage, general-purpose computation, and execution of system- and user-level applications [1], [3], [7]. *No* provision is made for local application execution and storage—laptops, palmtops, PDAs, and PIMs fundamentally differ from the InfoPad in this respect.

The user device, the InfoPad, consists of a radio modem, notebook-sized display, a pen pointing device, and video and audio input/output. The radio modem bandwidths are asymmetric, reflecting the importance of network information retrieval, with higher bandwidth (1-2 Mbits/sec per user) connectivity from the supporting backbone network

- *T.E. Truman is with Lucent Tchnologies, 791 Holmdel-Keyport Rd., Room M-230A, Holmdel, NJ 07733. E-mail: ttruman@tesla.ho.lucent.com.*
- *T. Pering, R. Doering, and T.W. Brodersen are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.*
  E-mail: {pering, rogerd}@eecs.berkeley.edu.

Manuscript received 8 Apr. 1997.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 104806.

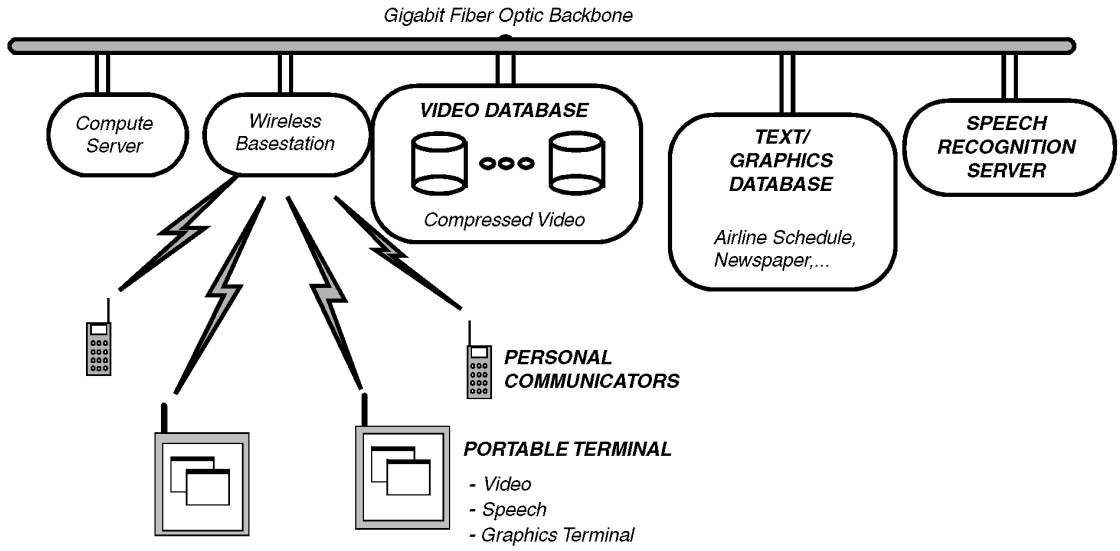


Fig. 1. Future infrastructure for information access.

to the terminal (downlink) and lower bandwidth connectivity (64-128 kbits/sec) in the reverse, uplink direction.

The InfoPad system architecture allows dramatic simplifications in both the actual hardware which is used, as well as the software and system management. A brief summary of some of the most important advantages are as follows:

- ***Reduced cost, complexity, and energy consumption:*** Moving the general purpose computing resources out of the portable device maximally reduces the cost, weight, and energy consumption by eliminating mass storage, high performance processors, and memories. Energy consumption for specific communication or I/O functions can be reduced by several orders of magnitude by replacing general-purpose computation with dedicated architectures.
- ***Ease of use and remote system support:*** The support for sophisticated applications and operating systems are provided by remote network managers. In this respect, the use model is closer to that provided by the telecommunications industry, in which the user I/O device, the telephone, has little complexity and the network providers perform system support and maintenance.
- ***Appearance of unlimited storage and computational resources:*** Since applications and server processes run on servers on the backbone network, it is possible to run sophisticated applications and computationally-intensive I/O algorithms—speech and handwriting recognition, for example—without the cost or energy consumption incurred in providing local high performance computation. Similarly, mass data storage is provided by storage and application servers rather than in local disks or flash memory.

This paper focuses on the architecture and implementation of the InfoPad: a portable information access device that supports real-time access to an infrastructure of multimedia information services via a high-bandwidth, low-latency wireless link.

The following section describes the hardware architecture of the InfoPad, which supports the above stated goals and associated internal protocols. This is followed by an evaluation on the critical characteristics along with directions for future developments.

2 Architectural Optimizations for Energy-Efficient Wireless Access to Multimedia Information

Externally, the InfoPad terminal provides a pen- and speech-based user interface to applications, along with a graphics and full-motion video display (Fig. 2). Since the link to the network is central to the operation of the device, a natural model for the portable device is that it is simply a multimedia-enabled extension of the backbone network. Several architectural features distinguish the InfoPad from desktop, notebook, and network computers, as well as from PDAs and PIMs. These features are outlined below.

2.1 Peripheral vs. Central Processing Unit

The InfoPad differs in a fundamental way from other portable computing platforms—including laptops and handheld personal information devices—in that *local execution of end-user applications is not supported.* The InfoPad system architecture embodies the logical extreme of thin-client computing.

Experience with an earlier prototype [3] indicated that the microprocessor subsystem, which was responsible for managing data transfers between the wireless modem and the I/O-processing chipset, consumed a significant fraction of the overall power budget and was also a primary performance bottleneck. The most delay-sensitive activities, such as moving the pen and expecting the cursor to track location in real-time, typically generate a large number of very small data transfers, so that the microprocessor spends the majority of its cycles entering and exiting interrupt service routines and setting up data transfers. Due to the delay-sensitivity and asynchronous nature of these transfers,

Fig. 2. The InfoPad portable multimedia terminal.

it is difficult to amortize the transfer setup overhead over more than one or two I/O packets.

Further, performing complex multimedia-data processing functions in dedicated hardware that is optimized for energy-efficient operation reduces the energy-per-operation by several orders of magnitude relative to software. Conventional general-purpose processors (e.g., Alpha, Pentium) focus entirely on instruction-per-second metrics, and typically require 100 milliwatts/MIP; energy-optimized, general-purpose processors such as the Strong Arm require 1-10 milliwatts/MIP. Fully dedicated, optimized hardware, on the other hand, typically requires less than 0.01 milliwatts/MIP.

Thus, for portable devices, there is a strong incentive to completely eliminate the microprocessor subsystem in favor of fully dedicated hardware. However, the disadvantage of dedicated hardware is the lack of flexibility and programmability and, for research purposes, it was desirable to have the ability to develop and test new algorithms and protocols.

The requirements outlined above lead to an optimization in which the user accessible central processing unit (CPU) is functionally removed from the architecture of the portable and networked based resources are used instead. Unlike a local CPU architecture, in which I/O peripherals enhance the functionality of the core processor, our goal was to design intelligent peripherals that are capable of processing I/O events and can manage data transfers without relying on a centralized processor.

The general-purpose processing unit is viewed as simply another peripheral subsystem that exists to complement the functionality of the I/O peripherals, thus the processor is more appropriately termed a *peripheral* processing unit (PPU). Our goal was to reduce the role of this PPU to be that of a simple controller that, in normal operation, initializes the system and handles complex protocol processing (e.g., handoff requests) that are most easily implemented in software.

2.2 Class-Based Communications Protocols

A second fundamental difference that separates the InfoPad from other mobile computing and information access devices is that it is primarily a *communications* device that is optimized for *energy-efficient* access to *multimedia* data in an indoor *wireless* environment. Thus, the algorithms, protocols, architecture, and components are all designed with a bias toward this specific context.

Throughout the InfoPad architecture, it was necessary to distinguish between classes of data, where each class has its own service needs, primarily because of the required support of interactive multimedia over a wireless network, where bandwidth is limited and reliability can vary dramatically. The heterogeneous mix of traffic supported over the wireless link requires that the link level protocol be aware of the delay and reliability requirements of a particular packet, and tailor the behavior of the protocol to meet the requirements of each class of traffic.

For example, in a vector-quantized image compression scheme, it is possible to differentiate the quantization codebook from the quantized image. Since the codebook is relatively small and will be used to decode many image frames, increasing the reliability of the codebook transmission (via forward error correction coding, retransmission, or a combination of both) has little impact on the overall bandwidth requirements but has a dramatic impact on the overall quality of the decompressed image. Data frames, which require more bandwidth and are more delay-sensitive, can be transmitted with little or no error correction: Corrupted image frames with bit-error rates of up to still provide the viewer with a good idea of the overall image composition [4].

2.3 Low-Overhead, Minimal-State Communications

Consistent with the goal of exploiting network resources, the amount of state maintained in the portable device is minimized and, for this reason, *explicit* support for end-to-end transport and internetworking protocols over the wireless link is avoided.

As we will discuss in detail in the following section, optimizing the portable device for energy-efficient operation presented a strong bias toward implementing the majority of the system in custom hardware and, to the extent possible, eliminating the role of the microprocessor as the central functional unit. Since the portable device was so heavily dependent upon the wireless link, a foremost priority was to eliminate the dependency on software-based protocol stacks. The resulting architecture of the portable device can be viewed as a packet-routing system that supports data transfer between I/O subsystems and the wireless link entirely in hardware (i.e., without microprocessor intervention). Thus, it was necessary to examine the merits of supporting fully general protocols, such as TCP and IP, in a hardware system.

Since applications execute on the backbone network and general-purpose network connections between applications exist entirely within the backbone infrastructure, the mobile is relieved from the task of handling "standard" protocols. Often, these protocols are designed for generality, and either require superfluous fields in the protocol data structures or exhibit behavior that is unsuitable

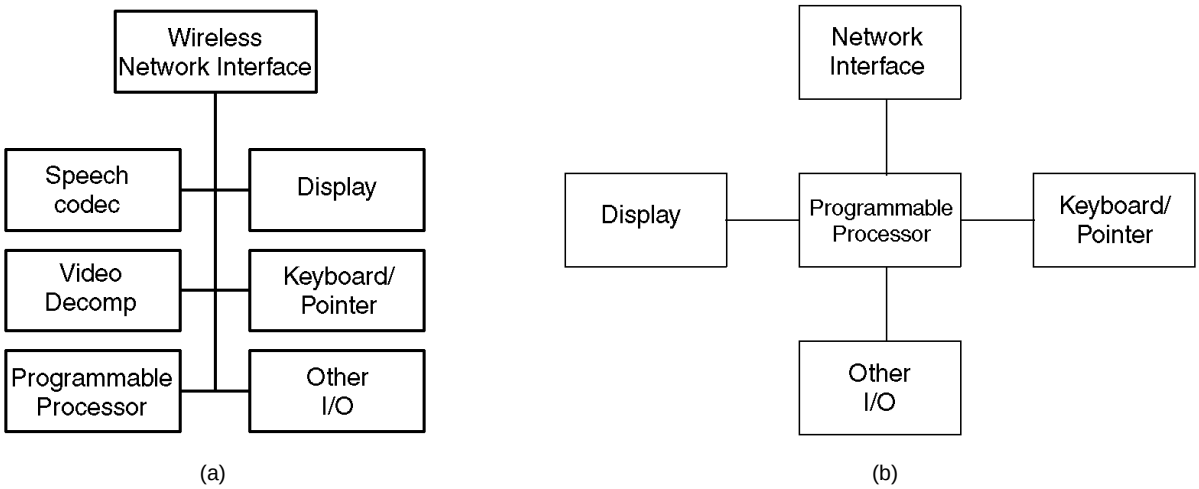


Fig. 3. Comparison of control organization of the (a) InfoPad and (b) classical computer architecture.

for use on error-prone wireless links. Terminating connection-oriented protocols at the basestation makes it possible to view the wireless link as a single-hop, point-to-point forwarding link.

Within the backbone network, a connection-oriented network that supports quality-of-service is assumed and, since the bandwidth requirements for the device are known in advance, it is possible to reserve the required network resources (bandwidth, server CPU cycles, etc.) at the time a connection is established. The InfoNet protocol suite [2] provides transport- and network-layer protocol services between the applications, servers, and basestations.

2.4 An Infrastructure for Mobile Multimedia Computing Research

An overarching design philosophy is that the InfoPad should be both a fully functional system and a research platform. In order to evaluate and test the computing model described above, it was necessary to design and build the entire system, from low-level, energy-efficient circuits to the infrastructure of network server and applications resources.

Thus, a primary goal of this work was to enable further research in the design of interactive applications, multimodal user interfaces, source/channel coding algorithms, network protocols, portable computing devices, and basestation architectures that are specifically targeted to wireless multimedia.

With this in mind, the remainder of the paper focuses on the architecture of the InfoPad multimedia terminal.

3 Architecture of the InfoPad Multimedia Terminal

Traditionally, personal computer systems are designed with a programmable processor at the core of the system (Fig. 3a). Since, in these systems, the primary system task is executing software applications, without the central processing unit the system would be useless. Peripheral I/O devices are added to support or enhance the functionality of the CPU core. When optimizing for the common case, system optimizations generally focus on increasing the fraction of time this central processing unit is able to give to applications.

Since the InfoPad system is designed to support a significantly different model of computing—one that is information access-centric—it is natural that the logical organization of the architecture is significantly different. The implementation of the InfoPad centers around the model of parallel I/O processing modules connected to a backbone network via a single-hop wireless link, rather than a central processing unit supported by peripheral I/O systems. With the long-term vision that the backbone network would exist within a virtual circuit-switched framework, our design philosophy was that the InfoPad should be an extension of the backbone network. Thus, the main function of the hardware is to support dataflow between the wireless link and multimedia sources or sinks.

Fig. 3b illustrates this organization. Foremost in the architecture is the wireless network interface, which supports full-duplex communications using both a 625 Kbit/s modem in the 2.4 GHz ISM band (downlink) and a 242 Kbit/s modem in the 920 MHz band (uplink). Other I/O subsystems are designed to autonomously interact with the network interface without support from the programmable processing unit. Data received on the downlink is transferred directly to the video, audio, and graphics subsystems without the assistance of the processor. Similarly, speech or pen input from the user is transferred directly to the uplink interface.

Conceptually, the architecture is analogous to an output-buffered, self-routing packet switch. At run-time, each data source is given a *type*-tag[1] which is used to identify the type of data generated (e.g., pen vs. speech) and, for each {*data source, type tag*} pair, a unique device destination address is assigned. When a source has data of a particular type available, it uses the *type* tag to dynamically determine the corresponding sink to which the data should be sent. Thus, once initialization is complete, data transfers between source and sink are autonomous, requiring no microprocessor intervention.

The *type* tag provides a mechanism to support lightweight protocols that provide data-specific transport services. For

1. The assignment of tags to a particular data class is globally shared with the backbone network software.

example, the transmitter interface module uses the tag to determine how the current packet is to be encapsulated, since optional fields such as packet length, forward error correction, or sequence number, may be omitted for certain types. Similarly, the receiver interface uses the *type* tag to decode a packet and to determine the destination of the incoming data (e.g., video frame buffer vs. audio codec interface). Using this mechanism, physical and link-level protocols can adaptively select what level of service a given packet requires by changing its *type*, or by changing the *type*-specific packetization options associated with that tag (to be discussed in Section 3.2). At a higher level, protocols supported by the InfoNet [2] gateway are able to utilize the *type* tag in making scheduling decisions.

3.1 Design Trade-Offs

Although the I/O devices were designed to operate autonomously, we chose not to eliminate the microprocessor from the design, primarily for the flexibility afforded by a general-purpose processor, for exploring these protocols is easiest in software. Since this does result in a less-than-optimal power budget, the system is designed to operate with the microprocessor providing only three support services: start-up initialization, packet scheduling for transmission over the wireless link, and support for link- and media-access protocols (including support for mobility).

A second power-related concession was made in the design of the wireless interface: The physical interface to the RF modems utilizes an commercially available FPGA to enable experimentation with both the wireless modem and the physical-layer protocols (e.g., FEC coding, clock recovery, etc.). Radio technology is rapidly advancing; and, while current radio technology is adequate for experimental designs, they do not provide the 2-10 Mbit/s envisioned for future devices. The FPGA design provides an interface which is easily changed to take advantage of new radios as they become available.

Overall, the primary technical challenge was to balance the low-power design against system flexibility (for use as a research tool) and system responsiveness (for actual use). In the following sections, as we discuss the specifics of the I/O subsystems, we will identify the design trade-offs and implementation choices that are driven by the architectural goals presented in Section 2.

3.2 IPbus Description

The core of the InfoPad hardware is a low-power bus, called the IPbus, dedicated to the movement of I/O data. Attached to this bus in a modular fashion are bus-mastering data sources and bus-slave sinks, as depicted in Fig. 4. Together, these I/O devices support two-way audio, pen input, monochrome graphics, and color video capabilities over a full-duplex wireless link; they are implemented as 10 full-custom ASICs in a 1.2-micron CMOS process. A microprocessor system is used to handle system initialization and higher-level protocol functions (e.g., collecting error statistics and signal strength measurements).

The IPbus is an 8-bus designed to run at a speed of 1MHz and a supply voltage of 1.2-1.5 Volts. This design provides a maximum throughput of 8 Mbit/s, well above

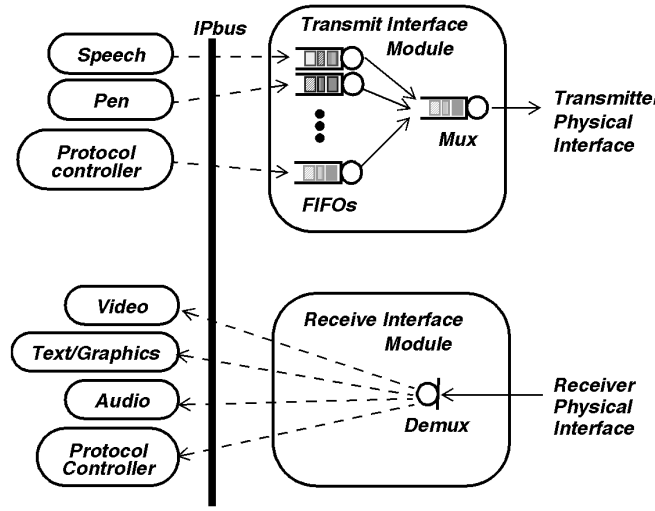


Fig. 4. Architectural organization of dataflow.

the 1 Mbit/s maximum supported by the radios, ensuring that the IPbus bandwidth is adequate for system dataflow. An 8-bit word size was chosen to minimize pin count, and hence package size, of the custom ASICs; a larger word size would not measurably improve system performance since the system throughput is constrained by the bandwidth of the radio channel.

The IPbus supports direct read/write transfers, as well as a packet-based transfer mechanism. Utilized only by the microprocessor subsystem, the direct read/write mechanism allows the processor to directly configure a device, query status, and respond to interrupt conditions. Packet-based transfers are used for interdevice communication: The source device indicates a new transfer by sending a start of packet (*SOP*) byte, followed by a variable number of data bytes, and terminates the transfer by sending an end of packet (*EOP*) byte to the sink device. Included in the *SOP* byte is the 6-bit *type* tag which identifies the data-type of the packet (e.g., pen, audio, etc.). The *EOP* byte contains additional (optional) status information which can be used to identify packets that are corrupted during transmission over the wireless link, for example.

Data transfers that are not required to be atomic: distinct data streams from different sources can be interleaved across the bus and simultaneous transfers to the same sink device by multiple sources are allowed. For example, it is possible for the audio, the pen, and the microprocessor to simultaneously transfer data to the transmitter interface. This removes the requirement for store-and-forward protocols in the I/O peripherals, decreasing the overall system delay, but requires that the sink devices be capable of demultiplexing the incoming data.

3.3 Wireless Interface Subsystem

In the early design phases (1992-1993), the dearth of high-speed wireless modems suitable for use in the InfoPad and the uncertainty that standard link-level protocols would provide adequate performance for multimedia over wireless, demanded reconfigurable wireless interface—one that was flexible in its ability to interface to a variety of wireless

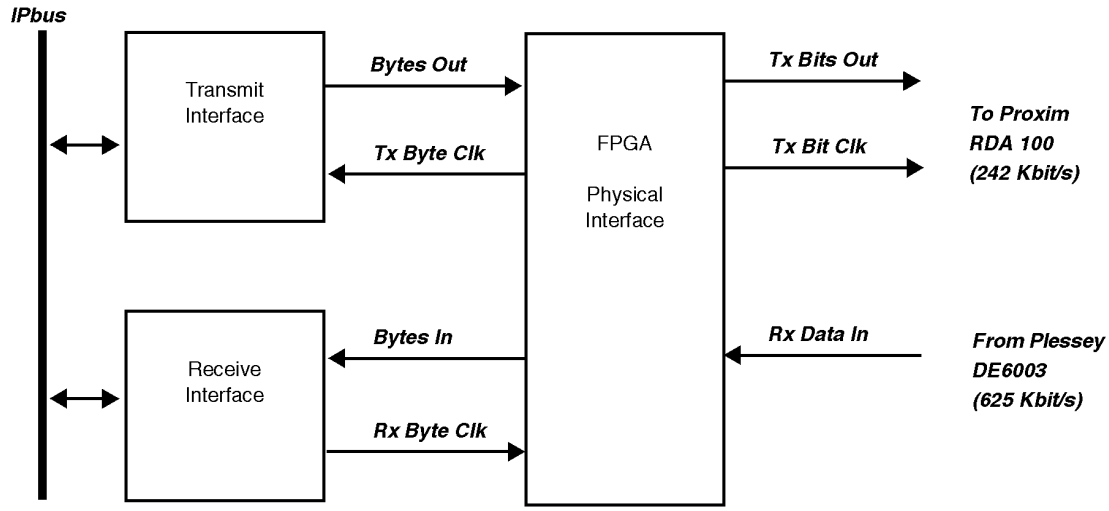


Fig. 5. Block diagram of wireless interface subsystem.

modems, as well as the ability to support experimentation with link and media access protocols. Our strategy was to partition the interface into three parts, shown in Fig. 5: a transmitter (TX) interface ASIC, a receiver (RX) interface ASIC, and a reconfigurable physical interface module, implemented in an FPGA. The TX and RX modules handle packet- and byte-oriented functions, while the FPGA provides bit-level manipulations, such as forward error correction (FEC) coding and the physical signalling interface to the wireless modems. This partitioning allows the underlying modem to change, requiring neither a change to byte- and packet-oriented operations nor an ASIC refabrication.

During the early design phases, the best commercially available radio modems suitable for use in the InfoPad terminal did not provide the 2-10 Mbits/sec envisioned for fully functional systems of the future. For this reason, to provide as much bandwidth as possible, separate uplink and downlink modems were utilized. The downlink modem, the Plessey DE6003, operates in the 2.4-2.5 GHz band and employs binary FSK modulation to provide a 625 Kbit/sec raw bit rate; this modem was designed for use in a slow frequency hopping system and has 100 available half-duplex frequency channels. The uplink modem, a Proxim RDA300, operates in the 902-926 MHz band and provides a 242 Kbit/sec raw data rate; four noninterfering, full rate and three noninterfering, half-rate (121 Kbits/sec) half-duplex channels are available.

Since the available bandwidth was below the ideal of 2-10 Mbits/sec, time-division multiplexing of the uplink and downlink channels was not considered for this prototype system. Instead, a simple frequency-division multiple access scheme was used within each picocell, where each cell is responsible for allocating a particular frequency band (i.e., channel) to each user in the system. Thus, the total number of InfoPads per cell in the prototype system was limited by the number of full-rate uplink channels to a maximum of four.

Several classes of data link protocols are supported in the InfoPad, corresponding to the level of reliability required. On the downlink, best-effort broadcast and multicast transmissions are supported, along with connection-oriented, variable-reliability service ranging from unacknowledged best-effort transmission to a Type I hybrid-ARQ [15] data transfer. With the exception of multicast transmission, the same data link protocols are supported on the uplink.

In the following subsections, we outline the salient features of the protocol support primitives.

3.3.1 Packet Structure

The basic packet structure supported over the wireless link is an extension of the IPbus packet format, shown in Fig. 6. The minimum overhead added by the wireless link is the single-byte pad alias, which is an address equivalent. Optionally, sequence number, packet length, and CRC fields may also be added. The inclusion of sequence numbers is a Pad-specific configuration parameter. The other optional fields are *type*-specific, giving a very fine granularity on how the link protocols treat particular classes of data, supporting our goal of providing lightweight, type-specific communications protocols.

3.3.2 Dynamic Network Addressing

Each InfoPad is assigned a unique identifier that is stored in nonvolatile memory and is presented to the backbone network to establish a connection. The backbone network uses this identifier to assigns a *pad alias*, which is a temporary, 1-byte, local network address used by both Pad and basestation indicate a radio packet's destination address. (Provision for multicast addressing is included and is discussed in Section 3.2).

3.3.3 Type-Specific Protocol Options

Many of the protocol primitives can be selectively enabled on a type-specific granularity. We outline these primitives below:

Variable error control: Multiple levels of error control and reliability effort are supported. At the lowest level of reliability, transmissions are unacknowledged and no error correction or error detection is employed; at the highest-level of reliability, a Type-1 Hybrid ARQ protocol is used.

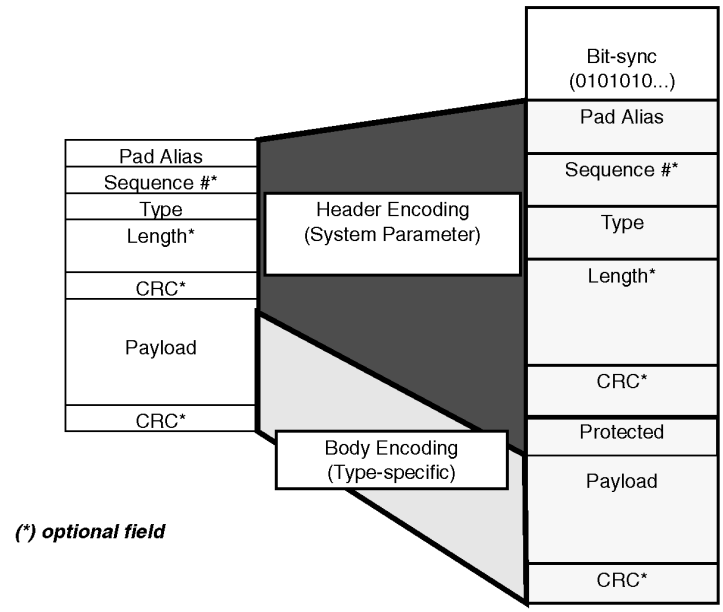


Fig. 6. Packet format.

Additionally, two different error correction codes[2] are available: a BCH (15, 5, 3) code, and a BCH (15, 11, 1) code. These variable-level encoding schemes allow bandwidth-intensive data (such as graphics or video) to be minimally encoded, while latency or content-critical data (such as video codebooks or pen packets) can be maximally encoded.

Differentiation of packet header and packet payload: An insight gained from experience with the earlier prototype is that, for many classes of data packets, there often are a few content-critical bytes which require higher reliability than the rest of the payload—a bitmap, for example, has an *x*- and *y*-coordinate followed by a series of pixel values, and displaying the bitmap at the correct location is far more critical than displaying every pixel value correctly.

For this reason, we chose to provide a mechanism that allows the link protocols to differentiate between packet header and packet payload: At a type-specific granularity, it is possible to encode the first 1-7 bytes of the payload with the same FEC coding as the packet header, while the remainder of the payload is encoded independently. An optional payload CRC field is available for data types that require correct transmission. In this way, it is possible to have the content-critical bytes maximally encoded, while the remainder of the payload is encoded at a completely different level. Interleaving is a standard mechanism for increasing the effectiveness of error correcting codes in the presence of burst errors [16].

Interleaving: The TX and RX subsystems provide a 15 × 16 interleaver / deinterleaver, which redistributes 240-bit blocks of FEC-encoded data into 16 15-byte blocks. In this configuration, we are able to correct a large number of burst errors of up to 48 bits in each 240-bit block.

In the remainder of this section, we outline the particular features of the wireless interface components.

2. An (n, k, t) error correcting block code uses *n* transmitted bits to send *k* information bits, and can correct up to *t* errors.

3.3.4 TX Interface

The TX subsystem is responsible for demultiplexing interleaved data streams (sent from different sources) and buffering them until they can be encapsulated and metered out to the FPGA interface. Data is encapsulated with the InfoPad radio packet format, which provides additional functionality such as error detection, length information, etc. Packet scheduling is accomplished in cooperation with the microprocessor subsystem.

The TX chip provides five distinct logical channels, each of which can handle one noninterleaved data stream. A single source per logical channel mapping is enforced in software. Associated with each logical channel is a ring buffer which provides storage for packets pending transmission (Fig. 7). Each ring buffer has a programmable number of entries—pointers to packets in an external memory—with up to 32 entries per ring. The number of buffers for each channel, as well as the size of each buffer, can be dynamically adjusted according to the type of traffic carried by the channel.

Since the link-level packet format is type-specific, the TX subsystem supports this functionality by optionally prepending *Pad alias*, *sequence number*, and *length* fields to each packet. At the start of each transmission, an internal lookup table—indexed by type—is consulted to determine which fields should be added to the current packet. In this way, the device provides a mechanism by which the link-level packet format can be dynamically adapted to the current transmission environment.

The architecture separates *packetization* from packet *scheduling*. For research purposes, the importance of this feature cannot be overstated, as it allows the processor to implement an arbitrary scheduling policy without requiring the processor to manage the transfers between peripheral I/O devices. Packet scheduling and link protocols are supervised by the microprocessor as follows:

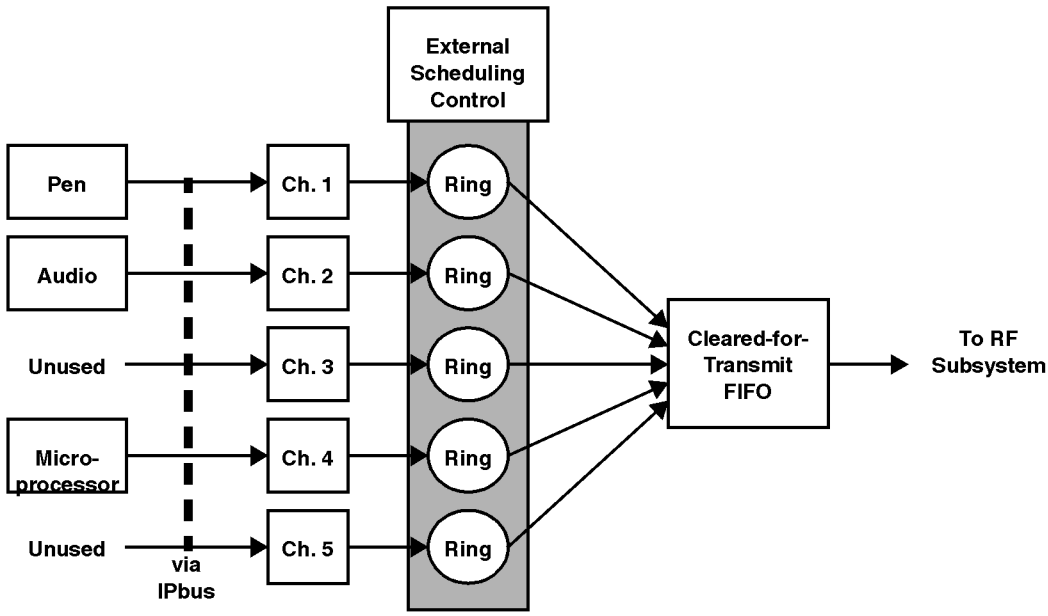


Fig. 7. Logical organization of TX buffering scheme.

Upon the receipt of a complete packet, the TX chip issues an interrupt to the processor subsystem. The processor reads the {*ring number, buffer number, packet type, length*} information from the device and records this information. At some time in the future, the processor may choose to queue this packet for transmission by pushing the {*ring number, buffer number*} to a "ready for transmit" FIFO in the TX module. At the completion of packet transmission, the TX chip again issues an interrupt, indicating that the packet represented by {*ring number, buffer number*} has completed transmission.

3.3.5 RX Interface

The RX subsystem is responsible for processing the incoming radio traffic. It processes the packet headers received from the FPGA and routes the data body to the appropriate destination. Packet headers can optionally be duplicated and forwarded to the local microprocessor for statistical monitoring, allowing the microprocessor to monitor the packet traffic, dropped packets, and passed/failed CRCs without processing or transporting body data.

The internal architecture of the device is a simple state machine that controls data flow to an internal FIFO. Packet destination is determined by a programmable lookup table based on the packet's *type* field which enables a variety of data flow scenarios. In normal operation, data is sent directly to the appropriate sink device, though for debugging, specific datatypes can be routed through the microprocessor, allowing monitoring, modifying, and rescheduling the packet before forwarding to its final destination.

The device performs only simple, pass-through routing: Once data is available in the receive FIFO, it is transferred out to the IPbus at the first opportunity. One disadvantage of this pipelined reception path is the inability of the receiver subsystem to drop incoming packets, since the payload CRC is not available until after the packet data has been forwarded on to the data sink. Therefore, data sinks that wish to discriminate between clean and corrupted data must buffer incoming bytes before processing can proceed.

In addition to supporting point-to-point addressing via the *pad alias*, the RX interface supports both multicast and broadcast addressing. A second alias, the *group alias*, is used to identify a particular multicast group; incoming packets with an Alias field matching either the *pad alias* register or the *group alias* register are accepted by the receiver module. With the exception of broadcast packets, all other packets are ignored.

3.3.6 FPGA Interface

The FPGA subsystem is the intermediary between the RX or TX subsystems and the physical radios. In addition to providing the error correction coding and CRC modules, the primary responsibilities are outlined below:

Signalling interface: Since there is no standard physical interface to wireless modems, supporting multiple physical interfaces is a task which is particularly well suited to programmable logic. Virtually all wireless modems support a common set of control signals, such as power up/down, channel selection, transmit/receive selection. The FPGA module provides an abstraction of the underlying mechanisms so that the RX, TX, and processor modules have a uniform view of the wireless modem primitives.

Timing and data recovery: On the downlink, the DE6003 modem interface presents only the raw received data signal—i.e., the analog output of a hard-limiting comparator (binary FSK modulation is used). This signal, which is has a nominal bit rate of 625 Kbits/sec, is oversampled by a factor of 16 (10 MHz), and is used to recover both the timing information and data sequence. At the start of a transmission, a sequence of 48 alternating 1s and 0s is sent, followed by a 32-bit framing character; once this is received, the tracking feedback loop is opened for the duration of the packet. Due to an implementation limitation of these

modems, the maximum continuous transmission is 10 milliseconds, so that clock drift between transmitter and receiver during a packet time is small enough to ignore.[3]

Real-time interrupt: To support real-time protocol requirements (e.g., frequency hopping), a real-time interrupt is provided with up to a five resolution. For reservation-based media access, the guard interval between channel uses is inversely proportional to the accuracy of the reservation timer. Placing the timer block near the receive path provides a convenient mechanism for accurately (within a bit-time) synchronizing the mobile with the basestation.

3.4 Microprocessor Subsystem

The microprocessor is responsible for system initialization and various high-level protocols. The system is based around an ARM60 microprocessor running at 10 MHz (i.e., 10 MIPs) with 512 KB of RAM, with 128 KB ROM for program storage. No cache or floating point unit is present in this microprocessor.

To maintain power efficiency, the microprocessor system is designed with hardware support for a software-initiated idle state. When the software determines that it has no more work to be done, i.e., when waiting for some external stimulus, it signals an external controller to initiate the idle state. This controller gates the system clock, freezing the processor in midcycle and driving its power consumption to a minimum. The processor interrupt line is monitored by the controller, and with the next interrupt (e.g., a timer event or incoming data), the controller reactivates the processor clock.

3.5 Microprocessor Interface Chip

The processor interface (ARMIF) device is the bridge between the microprocessor bus and the IPbus, and its main roles are buffering data, byte-to-word conversion, and performing miscellaneous control functions. In the active mode, the *Master* channel directs data from the processor to the peripheral I/O chips (video, text/graphics, audio, transmitter), while the *Slave* channel collects packets from the chipset (pen, speech, and radio transmitter). A third channel, the *Direct* read/ write channel, provides an unbuffered read and write mechanism so that the processor is able to program the control registers and read back the status registers of the peripheral chips.

3.6 User Interface I/O Peripherals

3.6.1 Graphics Subsystem

The graphics subsystem is the primary output device for the InfoPad system. It consists of a low-power SRAM frame buffer (described fully in [7]), a controller module, and 640 × 480 monochrome LCD. Graphics operations (e.g., line drawing and text display) are performed in the backbone network in the *graphics server* [2], and the resulting bitmaps are sent directly over the wireless link and are rendered by the controller module.

Three shapes of bitmaps are supported: rectangular block, horizontal line, and vertical swath (32 bits wide, variable height.) Normally, received bitmaps are displayed regardless of the correctness of their content; if bit errors are incurred during wireless transmission, then the data is displayed with errors. This design choice was driven by the desire to maintain a responsive interactive user interface with bit error rates above .

A second mode, called protected mode, queues incoming bitmap data and displays it only if it the packet payload passes CRC. This is used in conjunction with a technique known as asymptotically reliable transmission [4], [5]. To maintain the responsiveness of the user interface, an initial, possibly corrupted, version of a bitmap is rendered as quickly as possible; in the background, the graphics server follows the initial transmission with low-priority[4], protected-mode update packets that cyclically refresh the entire screen. In this scheme, corrupted packets that were previously displayed are eventually replaced by either a clean refresh image or an entirely new image (again, possibly corrupted). This approach provides a very responsive interactive feel while providing a means by which, asymptotically, the screen image can be rendered without errors.

3.6.2 Pen Subsystem

The pen interface utilizes a commercially available digitizer tablet, which is attached to the underside of the graphics LCD panel. This digitizer feeds pen coordinates and button status to a custom ASIC, which provides a buffered interface to the IPbus. With the first available byte of pen data, the ASIC initiates an IPBus transfer to the *Target Address*, followed by a programmable number of data bytes; this configuration allows the system software to fine-tune the buffer size in order to balance time-to-transmit delay against the overhead incurred by sending very short packets. (The time-to-transmit delay is approximately one millisecond per byte, and the target round-trip delay is 30 milliseconds or less. In the current implementation, the default pen packet size is five bytes).

3.6.3 Audio Subsystem

The audio subsystem performs bidirectional audio buffering and provides a physical interface to a commercial codec, amplifier, and speaker. The audio channel supports eight KHz 8-bit-law encoded audio, which presents the wireless link with 64 Kbit/s raw audio bandwidth. Downlink audio is buffered in a one Kbyte FIFO, smoothing the delay jitter in the incoming audio packets, and is metered out of this FIFO at the 64 Kbit/sec rate. Uplink audio is generated at the same 64 Kbit/sec rate and transferred to its destination (typically the TX interface) via the IPBus.

In the current implementation of the InfoNet and the InfoPad, the uplink audio and downlink audio streams are handled as separate entities; further, audio streams for separate users are handled as separate entities as well. Thus, applications that require synchronization between the uplink and downlink streams, or between multiple users, are not explicitly supported. However, extensions to the the audio server, along with specialized applications or servers, could in principle provide the support for these application.

3. The 20 MHz system clocks are accurate to ± 20 PPM

4. These packets can be transmitted at lowest priority utilizing otherwise unused transmission bandwidth.

3.6.4 Video Interface

The video subsystem supports full-motion color video. A custom ASIC implementation consisting of five decompression chips plus four custom, low-power frame buffer ASICs, drives the external add-on color display [7]. Ideally, the video display and graphics display would be combined, reducing system complexity; however, during the design phase, lightweight, thin, low-power, color LCDs were unavailable.

As detailed in [7], video data is transmitted using an adaptive vector quantization compression scheme which divides the compressed information into two distinct types—video *data* and video *codebooks*—each of which has differing transport demands. The compression scheme used can deliver up to 30 frames/second over the wireless link.

4 Evaluation and Measurements

Central to the InfoPad computing model is the premise that backbone network resources (e.g., computational power and network bandwidth) will become virtually unlimited due to technology improvements (Moore's Law). It is also assumed that the quality of a 1-2 Mbit/sec indoor link can be provided and maintained consistently throughout an indoor environment, freeing users to roam within a building. A third assumption is that the trend in computer use is toward real-time information access rather than heavy comptutation.

It is necessary to keep these three assumptions in mind as we evaluate the architecture and implementation of the InfoPad *terminal*. As with all large systems-building research projects, certain facets of the system are implemented with the sole purpose of demonstrating the larger system concept, and thus leave room for further work. For this reason, we focus on the evaluation of the multimedia terminal with the assumption that the our three key premises hold.

The implementation of the InfoPad terminal described in the proceeding section uses a combination of full-custom ASICs and commercially available components to provide the required functionality and demonstrate the viability of the architecture. In several places, providing this functionality with commercial components came at a significant increase in the power budget.

However, the lack of available components (or weaknesses in the underlying technology) has fueled further research efforts to close the gap. Energy efficient microprocessor design [11], low-power, energy-efficient DC-DC conversion [10], fully integrated, CDMA transmitter and receiver implemented in CMOS [9], and circuit design for energy-efficient reconfigurable logic devices are several of the complementary research projects which were spawned from the InfoPad project.

In this section, we evaluate the strengths and weaknesses of the design, making a distinction between architecture limitations and implementation-specific limitations. We begin with a power breakdown by subsystem.

4.1 Architectural Evaluation

As a preface to an architectural evaluation, we return to the fundamental assumption about the role of the terminal in the overall system: that the portable terminal is a remote interface to networked I/O servers where the interface devices generate data at a predetermined maximum rate. That is, the I/O subsystems on the portable terminal and the I/O servers on the backbone network collectively determine a throughput constraint[5] that the IPBus and the wireless link must support. Once this throughput constraint is met, there is no advantage to making the device "faster."

Hence, data throughput or computational performance alone is not a useful characterization of the system. Assuming that the backbone network has adequate computation resources, user-level application "performance" becomes a function of the latency inherent in the remote-I/O architecture. As described in [2], round-trip latencies within the backbone network (from basestation to servers and back to basestation) were typically between 10-15 milliseconds, with occasional latencies of 20 milliseconds.

In the following subsections, our evaluation instead focuses on how well the internal architecture supported the mobile computing paradigm of remote-I/O via a wireless link.

4.1.1 Processor Utilization

One of the primary goals was to push the model of thin-client computing to its logical extreme, in order to minimize—ideally eliminate—the role of the microprocessor subsystem in the overall design, in order to reduce power consumption. Thus, it is of interest to evaluate the dependence on the microprocessor.

While active, the processor spends the majority of its cycles servicing the TX module, which has only a one MHz read/write interface. Waiting for I/O peripherals while responding to *packet-ready* notifications, clearing packets for transmission, and responding to *transmission-complete* notifications dominate the time that the processor subsystem is not in sleep mode.

Fig. 8 illustrates that of the nonvideo power budget; the microprocessor subsystem accounts for 20 percent of the power consumption (approximately 1.4 Watts) in fully active operation (100 percent duty cycle). However, since the microprocessor subsystem is composed of fully-static CMOS components, gating the clock reduces the power consumption to approximately 0.25 Watts (power due to clock distribution only). During normal operation of the InfoPad, the measured duty cycle shows that the processor is active 7 percent of the time when running at 10 MHz; this yields an average power consumption of 0.33 Watts.

A new approach to microprocessor design that uses a technique called *dynamic voltage scaling* (DVS), [17] promises to reduce the power consumption of this subsystem even further. DVS takes advantage of the square-law reduction in power consumption that comes from reducing the supply voltage for a CMOS circuit.[6] Since decreasing the supply voltage reduces the switching speed of the circuit, it is necessary to simultaneously decrease the operating clock frequency.

5. Clearly, an upper bound on the throughput constraint is the raw available bandwidth of the wireless link

6. The energy per operation of a CMOS circuit is given by $E = CV^2$.

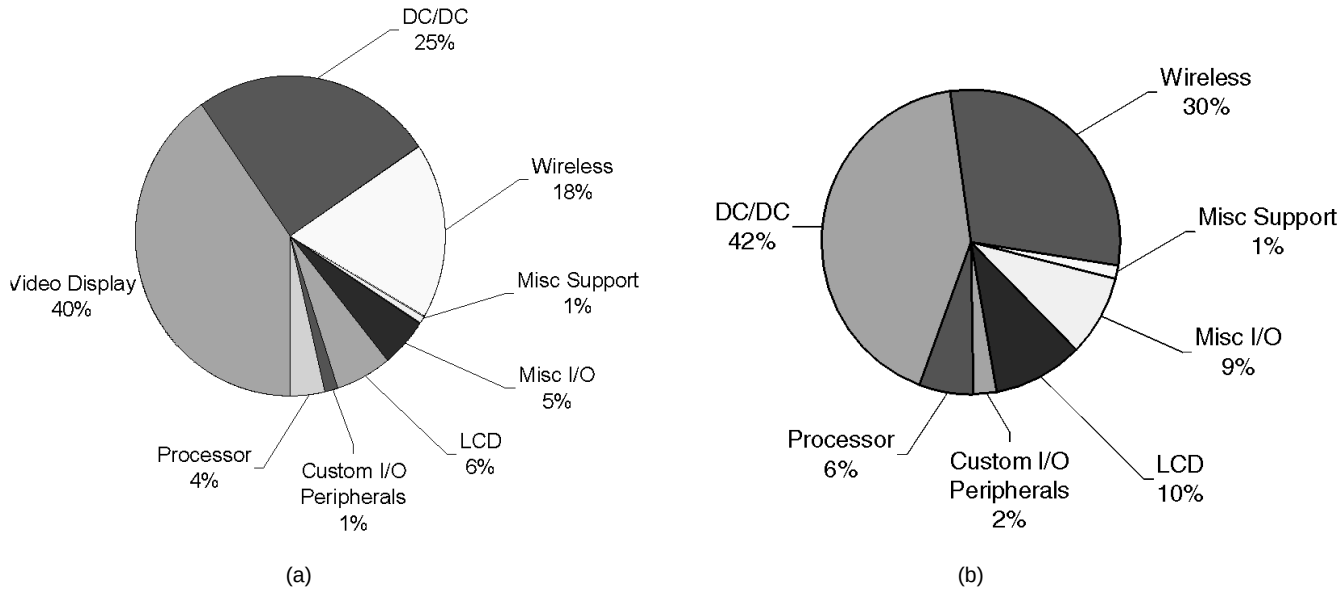


Fig. 8. Power breakdown by subsystem: (a) with color video display, (b) without color video display.

Scaling the processor clock with its supply voltage according to processor workload provides the opportunity to trade energy-efficiency against instantaneous MIPS. Further gains can be obtained by a tight integration of the processor subsystem components (e.g., DMA controllers, address decoding, etc.) onto the CPU core. Finally, by using optimized energy-efficient memories together with a low voltage-swing bus between the processor, it is expected that the entire processor subsystem will consume one to three milliwatts/MIP—an improvement of two orders of magnitude over the existing 250 milliwatts/MIP system [17].

4.1.2 Remote-I/O Processing Latency

A critical metric of the usefulness of the remote I/O architecture is the round-trip latency incurred as a packet moves through successive stages in the system. While the dominant source of latency is the network interfaces on the backbone network, early measurements ([2], [3], [8]) using standard workstations attached to a 10 Mbit/s Ethernet backbone demonstrate that a 30 millisecond round-trip latency was an achievable design constraint for a LAN-based backbone. This goal is based on the graphics refresh interval and gives the user an imperceptible difference between local- and remote-I/O processing for the pen-based user interface. Given this constraint, it is useful to evaluate the processing latency introduced by the interface between the IPbus peripherals and the wireless link. We break this latency into the following three components:

Packet generation: **3 microseconds**. This is defined to be the time elapsed from when the last byte of available uplink data until the packet is reported ready (i.e., a request for scheduling is generated). The bus-mastering architecture of the IPbus provides a direct path from each data source to the wireless network interface (via the TX chip buffers) without involving the processor. Thus, the packet generation latency is typically less than three IPbus clock cycles.

Scheduling: **160 microseconds**. This is the time required to process the scheduling request and clear the packet for transmission. To facilitate experimentation with a variety of scheduling algorithms and media-access protocols, packetization and scheduling are separated. Partitioning these functions into physically separate units increases the complexity of the packetizer by requiring it to support random access to available packets. This partitioning also increases intermodule communication by requiring the packetizer to interact with the scheduler several times for each packet, and each interaction requires several bus transactions.

The current implementation, with an idle transmitter and an empty transmit queue, has a worst case time on the order of 160 microseconds—50 microseconds to notify the processor, 10 microseconds for the processor to clear the packet for transmission, and 100 microseconds for the first bit of data (after 64 bits of synchronization preamble) to be transmitted over the wireless link.

Packet distribution: **1 microsecond**. This is defined as the time elapsed from the moment the first byte of available downlink data is ready until the first byte of the packet is sent to its destination device (e.g., pen, audio, etc.). Since the architecture employs direct, unbuffered routing from source to destination, the packet distribution latency is simply the time required to determine the hardware destination address for the given type, which can be accomplished in a single IPbus clock cycle.

The sum of these three components is 164 microseconds. This latency is insignificant compared to the latency incurred in the backbone network: 10-20 *milli*seconds on a standard 10 Mbit/sec LAN (well within the upper bound 30 millisecond round-trip). It is expected that state-of-the-art high-bandwidth networks that support QoS will be able to scale to support the 50 users per cell envisioned.

4.1.3 Communications Protocol Support

Because the InfoPad architecture supports type-specific link protocols, it is possible to experiment with a variety of

protocols and, given the current transmission environment, to fine-tune the parameters of each protocol to best handle each type of data. By characterizing the typical traffic patterns for each data type—a characterization which may be performed either off-line or dynamically—it is possible to eliminate unused fields, and to optimize for the common case.

The data link protocol, which provides a unreliable point-to-point link over the wireless medium, relies on the 1-byte pad alias field to indicate the receive address, and on the type-tag field (one byte) to identify the data type of the current packet. Optionally, this layer includes any combination of the following: packet length (two bytes), sequence number (one byte), header CRC (one byte), and payload CRC (one byte). Relative to standard protocols for wireless transmissions, this 7-byte overhead is significantly less: On a 5-byte pen packet, for example, typically only the pad alias and two CRC fields are added, incurring a 37 percent overhead. For comparison, the IEEE 802.11 draft standard, which uses a 28-byte MAC frame header, requires 660 percent overhead.

4.2 Implementation Evaluation

4.2.1 Power Consumption

The totals for the power consumption, broken down by subsystem, are presented in Table 1 and are graphically summarized in Fig. 8. The figures indicates the maximum power consumption, which is measured when all subsystems are fully active (100 percent duty cycle). Complete with the video display module, the InfoPad consumes 9.6 Watts, an order of magnitude higher than the ideal power budget.

The external video display, however, was intended to demonstrate the feasibility of compressed full-motion video over a wireless link and to demonstrate the low-power decompression chipset. It is worthwhile to consider the system without the external color video display because it is such a large fraction of the power consumption (3.9 Watts) and since comparable color LCD panels that consume less than one Watt are now commercially available. In the discussion below, the we analyze the *nonvideo* power budget.

Without the video display, the inefficiencies of DC/DC conversion surprisingly dominates the total power dissipation. These standard, off-the-shelf converters typically operate at 60-70 percent efficiency, expending nearly 2.5 Watts (42 percent of the total power) in providing the required supply voltages. The need for efficient DC/DC conversion is clear: A 90 percent efficient voltage conversion, for example, reduces the 2.5 Watts currently dissipated to 0.9 Watts—a 25 percent reduction in the total InfoPad power budget (including add-on LCD screen).

To address this need, a new DC/DC conversion design methodology has been developed, and a proof-of-concept low-voltage prototype IC has been demonstrated which remedies many of the limitations of current-day solutions [10]. Smaller size and lower power systems are achieved through the highest levels of CMOS integration, together with higher operating frequencies and minimum-sized inductor selection. A synchronous rectifier, whose timing is controlled in a low-power DLL, enables nearly ideal soft-switching and efficient conduction, even at ultra-low output voltages. Typical converter efficiencies range from above 90 percent at full load and 1.5 V and above, to 80 percent at minimum current load and voltages as low as 200 mV.

TABLE 1
BREAKDOWN OF POWER CONSUMPTION BY SUBSYSTEM

Subsystem (TOTAL mW)	Component	Power (mW)
Radio (1700)	Downlink	550
	Uplink	525
	FPGA	600
	EEPROM	0
	A/D Converter	25
Processor Subsystem (1380 max, 330 typical with 7% duty cycle)	PAL	400
	512K SRAM	600
	Processor	120
	Clock Distribution	260
Custom I/O Peripherals (137)		137
Misc. I/O (Pen tablet, speaker, codec, etc.)		500
Text/Graphics Transflective LCD (550)		550
Video Display Module (optional add-on) (3850)	4" Color LCD	2580
	5DC -> 1000AC	425
	5DC -> -8DC	75
	9DC -> 5DC	770
DC/DC Conversion (2416)	9 -> 5	2350
	5 -> -5	30
	5 -> 1.5	36.7
Misc. Support Electronics		75
Debugging Aide (Serial Port UART, LEDS)		75
TOTAL:	w/o Video Display: 6.8 Watts	With Video Display: 9.6 Watts

The second largest power consumer is the wireless link subsystem: including the FPGA interface module (0.6 Watts), the uplink, and downlink radios (0.55 and 0.53 Watts, respectively), and the A/D converter for measuring received

signal strength (0.25 Watts). This accounts for 30 percent of the total power dissipation, and the desire to substantially reduce the power dissipation in this subsystem has fueled research both in low-power reconfigurable logic [12], and in low-power RF transceiver design [9].

In [9], a fully integrated, combined RF and baseband CDMA receiver implemented in 0.6 micron CMOS, was described, and the total power consumption was less than 30 milliwatts. It is believed that for an indoor pico-cellular environment, where the transmit power at the portable device can be reduced to 0 dBm, aggressive low-power optimizations and advanced CMOS technology would enable an uplink transmitter to be realized with a power budget of 100-200 milliwatts.

Finally, the I/O processing ASICs that perform the computationally intensive functionality consume only 137 milliwatts—2 percent of the system energy consumption (excluding the external color LCD display). Clearly, in light of the preceding discussion, further optimization of these components for energy-efficient operation will have an insignificant impact on the overall energy consumption.

Table 2 summarizes the projected power budget for an implementation that uses in due to application of new and emerging technologies. With the application of advanced wireless modem design, a dynamic voltage scaled processor subsystem, high-efficiency DC/DC conversion, and new LCD technologies, it is reasonable to expect that the InfoPad multimedia terminal could be implemented with a power budget of 1.3 milliwatts.

4.2.2 Form Factor

Including the battery pack, the InfoPad measures 11 inches by 12 inches, is 1.3 inches thick, and weighs 3.3 pounds (1.1 kg). An open-case view of the InfoPad is shown in Fig. 9. Fig. 10 graphically summarizes the contribution, by subsystem, to the weight and surface area of the device; the weight contributions are detailed in Table 3. The printed circuit board (PCB) required 12 layers to accomodate the required routing between ASICs.

Higher levels of integration is the most obvious way to improve the form factor. Preliminary estimates indicate that by using a 0.35-micron, three-metal CMOS process, it would be possible to combine all of the functions of the I/O-processing ASICs onto a single die, eliminating the 10 of the 11 ASICs. Reducing the number of high-pinout ASICs is beneficial in three ways:

1) Eliminates the weight of the chip packaging;
2) Reduces the surface area and, hence, weight of the PCB board;
3) Simplifies the PCB routing, and allows for a reduction in the number of layers in the board.

Together with the pinout and socket spacing, we estimate that at 30-40 percent of the existing surface area of the PCB, and 20-30 percent of the current weight. Pushing the higher integration between the analog and digital components is needed as well, given the large fraction of the overall board area that is utilized by discrete components.

TABLE 2
APPLICATION OF NEW TECHNOLOGY TO INFOPAD ARCHITECTURE

Subsystem	Current Power Consumption (mW)	New technology	After application of new technology (mW)
Radio subsystem	1700 mW	Eliminate FPGA Full-custom CDMA transceiver with multiuser detection	100 mW uplink (0 dBm output)
			50 mW downlink
10 MIP Processor + Main memory, I/O interfaces, main bus components	340 mW at 7% duty cycle, 5 Volts, 10 MHz (225 mW/MIP for entire subsystem)	Dynamic voltage scaling + energy- optimized ARM processor and subsystem 1.1 - 3.3 Volts 10 MHz, 1-3 mW/MIP	0.7 - 2.1 mW
DC/DC converters	2400 mW at 2.0 Volts 58% efficiency	High-efficiency, low-voltage DC/DC conversion at 1.1 Volts, 90% efficiency	200 mW
LCD panel	550 mW monochrome LCD	state-of-the-art 800x600 color LCD	800 mW
Misc I/O (standard RS- 232, debugging ports, *etc)*	500 mW	fully-optimize & eliminate debugging circuits	10 mW
Custom I/O processing ASICs	150 mW at 2.0 Volts	1.1 Volt, single-chip implementation	50 mW
Approx. Total	**5000 mW**		**1300 mW**

5 CONCLUSION

Optimizing the system architecture and design of the future "personal computer" for mobile wireless access to network based services requires a new relationship between local computation and network access capability. The InfoPad explores a design point where the terminal functions as a remote I/O interface to user-accessible computing, information, and storage resources that are removed from the portable device and are placed on a shared, high-speed backbone network of servers. This optimization allows the minimal cost, weight, and energy consumption.

Results show that, by using an optimized architecture for communications, along with low-power design techniques, high real-time multimedia data can be manipulated while requiring only a small fraction of the overall system power. Future research should focus on the other power consuming components which includes displays, high-efficiency DC/DC conversion, energy-efficient microprocessor de-

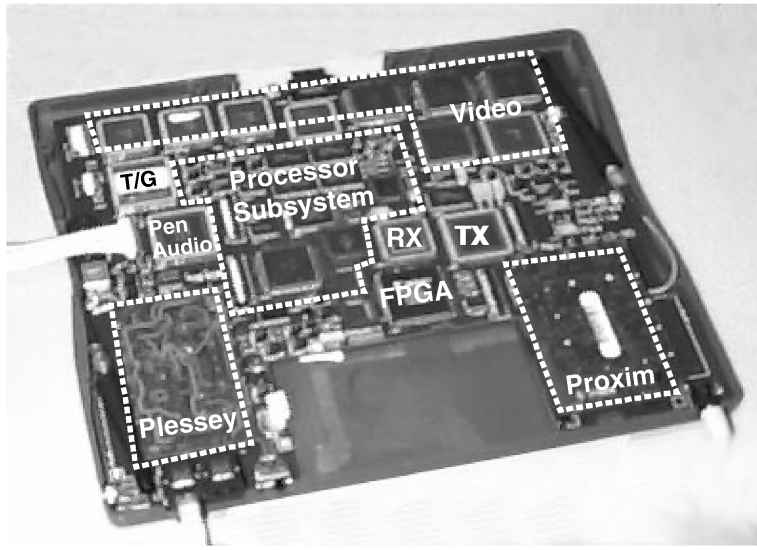


Fig. 9. Interior view of InfoPad terminal.

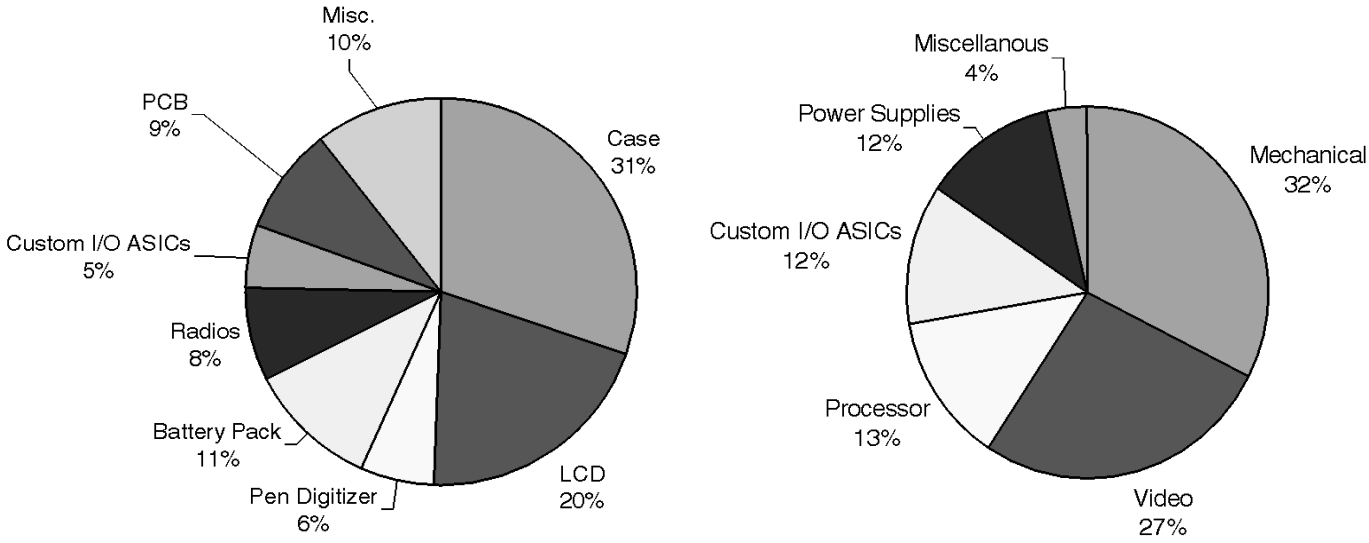


Fig. 10. Breakdown by subsystem of weight (a) and surface area utilization (b).

sign, fully integrated, low-power RF transceivers, and low-power programmable logic technologies.

ACKNOWLEDGMENTS

This work was supported by DARPA contract J-FBI-93-153.

REFERENCES

[1] S. Sheng, A. Chandrakasan, and R.W. Brodersen, "A Portable Multimedia Terminal," *IEEE Comm.*, vol. 30, no. 2, pp. 64-75, Dec. 1992.

[2] S. Narayanaswami et al., "Application and Network Support for InfoPad," *IEEE Personal Comm.*, vol. 3, no. 2, pp. 4-17, Apr. 1996.

[3] B. Barringer, T. Burd, F. Burghardt, A. Burstein, A. Chandrakasan, R. Doering, S. Narayanaswamy, T. Pering, B. Richards, T. Truman, J. Rabaey, and R. Brodersen, "InfoPad: A System Design for Portable Multimedia Access," *Proc. Calgary Wireless '94 Conf.*, July 1994.

[4] R. Han and D.G. Messerschmitt, "Asymptotically-Reliable Transport of Multimedia/Graphics Over Wireless Channels," *Proc. SPIE Multimedia Computing and Networking*, vol. 2,667, pp. 99-110, 1996.

[5] R. Han, "Progressive Delivery and Coding of Interactive Multimedia Over Wireless Channels," PhD dissertation, Univ. of California, Berkeley, ERL Memorandum #UCB/ERL M96/XX.

[6] A. Chandrakasan, "Low-Power Digital CMOS Design," PhD dissertation, Univ. of California, Berkeley, ERL Memorandum #UCB/ERL M94/65.

[7] A. Chandrakasan, A. Burstein, and R.W. Brodersen, "A Low-Power Chipset for a Portable Multimedia I/O Terminal," *IEEE J. Solid-State Circuits*, vol. 29, no. 12, pp. 1,415-1,428, Dec. 1994.

[8] M. Le, F. Burghart, S. Seshan, and J. Rabey, "InfoNet: The Networking Infrastructure for InfoPad," *Digest of Papers, IEEE COMPCON '95: Technologies for the Information Superhighway*, pp. 163-168, Mar. 1995.

[9] S. Sheng, L. Lynn, J. Peroulas, K. Stone, and R.W. Brodersen, "A Low-Power CMOS Chipset for Spread-Spectrum Communications," *Proc. Int'l Solid-State Circuits Conf.*, pp. 346-347, 471, San Francisco, 8-10 Feb. 1996.

TABLE 3
WEIGHT BREAKDOWN BY COMPONENTS

Component(s)	Weight (g)	% of Total	Comment
Battery	170	11%	5- 9V Batteries with connector strip
Case	482	30%	Injection-molded plastic
LCD Panel	326	20%	
Digitizer Board	99	6.2%	
Downlink radio	54	3.4%	Plessey DE6003
Uplink radio	71	4.4%	Proxim RDA 300
Custom I/O Processor ASICs	88	5.5%	15ASICs, ceramic flat-pack casing
PCB	142	8.7%	12-layer board
Processor Subsystem	43	2.6%	ARM60, EEPROM, address decoder PAL, UART, RJ-45 serial port connector
Miscellaneous	125	7.8%	Connectors, sockets, discrete components, etc.

[10] A. Stratakos, S. Sanders, and R. Brodersen, "A Low-Voltage CMOS DC-DC Converter for a Portable Battery-Operated System," *Proc. 1994 Power Electronics Specialist Conf.*, vol. 1, pp. 619-626, Taipei, Taiwan, June 1994.

[11] T. Burd and R. Brodersen, "Processor Design for Portable Systems," *IEEE J. VLSI Signal Processing*, to appear 1997.

[12] A. Abnous and J. Rabey, "Ultra-Low-Power Domain-Specific Multimedia Processors," *Proc. IEEE VLSI Signal Processing Workshop*, San Francisco, Oct. 1996.

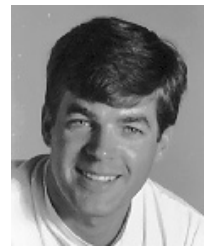
[13] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links," *Computer Comm. Rev. (ACM SIGCOMM '96 Conf.*, vol. 26, no. 4, pp. 256-69, Oct. 1996.

[14] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proc. IEEE*, vol. 28, no. 10, pp. 1,374-1,396.

[15] S. Lin and D. Costello, *Error Control Coding*, chapter 6. Englewood Cliffs, N.J.: Prentice Hall, 1983.

[16] S. Lin and D. Costello, *Error Control Coding*, chapter 9. Englewood Cliffs, N.J.: Prentice Hall, 1983.

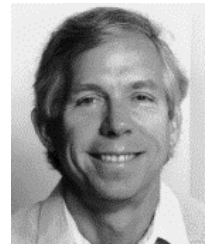
[17] T. Burd and R. Brodersen, "Processor Design for Portable Systems," *J. VLSI Signal Processing*, vol. 13, nos. 2/3, pp. 203-222, Aug./Sept. 1996.



Thomas E. Truman received the BS, MS, and PhD degrees from the University of California, Berkeley, in 1992, 1994, and 1998, respectively. From 1987-1992, he designed embedded systems software for automated test equipment for Pacific Western Systems in Mountain View, California. From 1997-1998, he was with Cadence Berkeley Labs and, since August 1998, he has been a member of the technical staff at Bell Labs in Holmdel, New Jersey. His research interests include hardware/software embedded systems, link- and MAC-layer protocols, formal protocol verification, and system-level design methodologies that span these fields.

Trevor Pering received the BS and MS degrees from the University of California, Berkeley, in 1993 and 1995, respectively. He is now a PhD candidate at UC Berkeley, where he is investigating energy efficient software, including operating systems for dynamic voltage scaled microprocessors.

Roger Doering received the BS degree from Case Western Reserve University in 1973, the MS degree from the University of California, Berkeley, in 1978, and is currently a doctoral candidate at UC Berkeley. His doctoral research involves the design and implementation of a single-chip, silicon micro-machined display ASIC for projection or mounting in a visor unit. He holds three U.S. Patents, with several others in progress. He co-authored the freshman-level text *Electrical Engineering Uncovered*. He is a member of the IEEE.



Robert Brodersen received the BS degree in electrical engineering and mathematics from California State Polytechnical University in 1966, the MS and Engineers degrees from Massachusetts Institute of Technology (MIT) in 1968, and the PhD degree in engineering from MIT in 1972. In 1972, he joined the Texas Instruments Central Research Lab. In 1976, he joined the faculty at the University of California, Berkeley, where he is currently a professor, holding the John R. Whinnery Chair in Electrical Engineering. He has received numerous best paper awards from conferences including ISSCC. He received the W.G. Baker Award for Best IEEE Publication in all areas in 1979. He became a fellow of the IEEE in 1982, received the IEEE Morris Liberman award for "Outstanding Contributions to an Emerging Technology" in 1983, and received Technical Achievement Awards from the IEEE Circuits and Systems Society in 1986 and from the IEEE Signal Processing Society in 1991. He became a member of the National Academy of Engineering in 1988.