Building Microelectronic Systems in a University Environment

John Poulton
Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina 27599-3175

Invited Presentation

This paper describes a ten-year-long technical and social experiment in our department to build a laboratory that designs and fabricates microelectronic systems in support of research in computer architecture. This experiment has been fairly successful by several measures: We have built and demonstrated a number of systems, some of significant size and complexity, spanning a fairly wide range of approaches and applications. We have validated the laboratory's original premise, demonstrating economies of scale by sharing a system-building facility over multiple projects. Most importantly, we have *fielded* experimental systems on a long-term, maintainable basis. Our Microelectronic Systems Laboratory (MSL) may serve as a useful model for others who want to develop system-building capability in a university setting. I'll describe the model in three parts:

1. An outline of the context, purpose, organization, and working style of the lab, along with a brief chronology and description of the experimental systems we have built.
2. A history of the longest-running and most ambitious project we have so far undertaken, building several generations of Pixel-Planes graphics systems.
3. A summary of what we believe we've done right, what we got wrong, what we could have done better, and few remarks about where we may go from here.

1 Laboratory Model

The MSL exists within the Computer Science department at UNC, so a few words about the department are needed to understand the lab's context. It is relatively small (currently 22 faculty and 120 graduate students) and is primarily involved in graduate education and research. There are no undergraduate majors, though about 30 Math BS's graduate each year with a computer science concentration. The department does first-class research in interactive computer graphics and is also fairly strong in microelectronic design, natural language processing, programming languages, and functional languages and machines. Founded as an autonomous department, it is not associated, for example, with electrical engineering (there are no traditional engineering curricula at Chapel Hill).

Fred Brooks, the department's founder, has strongly stamped the its research style. *Experimental* computer science is emphasized; several groups have built complete, full-scale systems and put them in the hands of users, typically from outside the department and often outside computer science. Brooks measures success by asking questions such as, "Does the system allow users to produce useful, publishable results

that couldn't be obtained any other way?" This style is especially appropriate for interactive graphics systems, both software and hardware. Where interactivity is essential, only a full-scale prototype can answer such questions.

During 1980-81, the confluence of the Mead and Conway VLSI revolution and a state initiative in microelectronics made it possible to consider building hardware systems, whereas before we had built mainly software systems. Brooks, then department chairman, felt strongly that prototype architectures based on silicon would be of value to us *only* if we could build full-scale *systems* out of custom chips. In early 1981, he created the Microelectronic Systems Laboratory (MSL) and hired its director, Vernon Chi, and three other lab staff (including the author). This act required considerable courage of conviction: the lab was launched using research faculty salary, supplied from the state microelectronics initiative, at a time when the department had only about ten faculty.

The MSL charter is to build running, reliable, maintainable hardware systems, based both on custom VLSI and on commercial technology, in support of research in computer architecture. The lab has no repair or maintenance responsibilities, other than maintaining (or making it possible to maintain) its own products and facilities, so it is quite different from a typical university electronics repair shop. At the beginning, the MSL had no organizational model, equipment, or facilities. Since the lab was built from scratch, the model for its organization and operation developed in a unique way.

People. The staff currently consists of four research faculty, four research associates, a laboratory manager, and a secretary. Most staff members have been with the lab for some years; several have worked on more than one project. All are self-starters, so there is little need (or spare cycles) for personnel management. The organization is collegial and has a matrix-management-like style. There is no hierarchy of engineers and technicians; staff engineers do their own assembly and fabrication. MSL does not function as an engineering services contractor, but rather as a set of facilities shared by various projects. Each of the lab's staff is a member of one or more research groups.

Funding. Initially, staff salaries (4 positions) were state-supported. The lab's first major capital equipment purchases were enabled by an NSF "Coordinated Experimental Research" (CER) grant, which launched many other projects in the department. Currently, the lab's operations are supported by an NSF "Institutional Infrastructure Program" (IIP) grant. These grants have also provided one or two staff salaries. The remainder of our facility and staff have been underwritten by funded research projects that use the laboratory.

Projects. Most projects are initiated by the research interests of a regular faculty member. Hardware architecture research groups typically involve one or more faculty, graduate students, and one or more MSL staff. Projects seem to have three phases: (1) ideas crystallize and proofs-of-concept are built; (2) architectures are developed to the point of implementation, and a full-scale prototype is designed and built; (3) the prototype is delivered and put into use, and the lessons learned from the development

exercise and from users' feedback are applied to the next research project. Having a shared laboratory facility greatly facilitates each of these phases. Small proof-of-concept projects, often required to demonstrate feasibility and to acquire funding for full-scale development, are usually built with scrounged resources and are often stymied by lack of facilities and experience. With permanent facilities and staff, MSL can do such projects rapidly and at low cost. Projects that build full-scale hardware prototypes enjoy access to superior equipment and CAD tools by virtue of the economies of scale of a shared facility, and the permanent staff accumulates know-how over multiple projects. Once a prototype is completed and fielded, the lab provides a long-term solution for maintenance.

A brief summary of the projects we have undertaken over the past 10 years shows their variety and scope; Figure 1 is a chronology of the lab's activity.

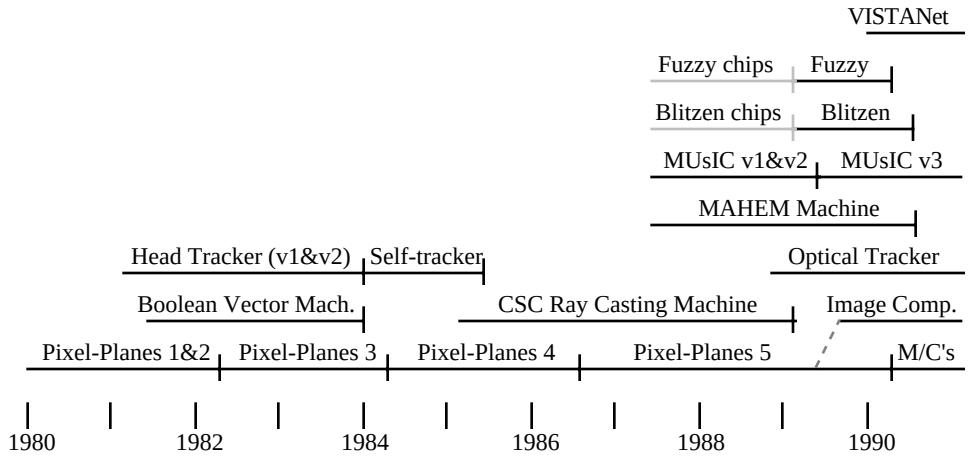

Figure 1: Chronological overview of projects in Microelectronic Systems Laboratory.

Two of our projects were collaborations with Duke University Computer Science. The **Boolean Vector Machine** (Robert Wagner [13]) is an array of bit-serial SIMD processors communicating on a cube-connected-cycles network. Wagner and I collaborated in designing several generations of custom chips (64 PE's, 5MHz, hot-clock nMOS), and John Austin of the MSL staff designed, built, and delivered an 8-chip system and a 1-chip test set, which was used for application and software development. Boards for two generations of **CSC Ray Casting Machine** were designed and built by Tom Lyerly in the MSL. These were based on architectures and custom silicon developed by Gershon Kedem and his group at Duke and at the Microelectronics Center of North Carolina (MCNC) [10]. The latest version of the machine, on 8 boards each with 40 custom chips, serves as a mechanical CAD modelling accelerator for a host workstation. Systems were delivered to Duke and to Cornell University, where many interesting applications have been developed. Work continues at Duke, where Computer Science and Electrical Engineering have joined forces to build their own systems laboratory.

Building useful head-mounted displays (HMD's) has been a challenging driving problem for much of our department's graphics research; we believe that satisfactory technical solutions are still some years away. We have been working both on fast computer image generation and on **Head Tracking**, measuring the position and orientation of the user's head in real time, so that images can be updated in the HMD in such a way as to appear stationary in the user's environment. Early prototype trackers involved head-mounted targets; version 1 had multiple strobed lights tracked by television cameras, while version 2 had a single light source tracked by lateral-effect diode cameras, with target rotation encoded in optical polarization. In his Ph.D. dissertation, Gary Bishop [1] developed ideas for an optical Self-Tracker chip, inspired by Xerox's optical mouse, that combined silicon sensors and image processing circuitry. Currently we are working on an optical tracker using tiny head-mounted cameras that measure the positions of multiple ceiling-mounted beacons. A small prototype has been completed; the goal is to have a real-time system in operation on a HMD by mid-1991.

The **MUsIC** project, a collaboration with Duke's Biomedical Engineering department, is building custom silicon digital delay lines for a phased-array ultrasound system that can image 3D volumes of order 10 cm on a side at 30 frames per second. The first generation chips were running by mid 1989, and a PC board holding eight of the devices is currently being integrated into a prototype imaging system. We are currently designing new, denser chips with mixed analog/digital functions, intended for a full-scale 3D display.

The **VISTANet** project will link multiple computing resources via gigabit networks to demonstrate real-time radiation cancer treatment planning. During treatment planning, a physician interactively manipulates and designs shapes and doses for radiation beams, superimposed on a 3D model of the patient's anatomy, typically derived from CAT images. Radiation dose contours will be computed on a remote supercomputer (a Cray Y-MP at the North Carolina Supercomputing Center). Updated dose data will then be shipped over the network to a graphics system (Pixel-Planes 5) at a second remote site, which will generate images of both the anatomical model and the dose contours according to the physician's desired viewpoint.

Several projects in the MSL have been one-board prototypes, rapidly designed and fabricated to serve as proof-of-concept demonstrations of new architectures. **MAHEM** is a hardware implementation of an adaptive histogram equalization (AHE) image processing technique developed by Steven Pizer [4] of our faculty. James Ericson, a student at UNC, built a machine based on inexpensive commercial parts in under a year. It performs AHE on medical images in about 1 second and is in regular use at UNC's Medical School. The **Fuzzy** inferencing engine prototype, also designed and built by a UNC-CS student, James Simon, contains a Fuzzy Logic Chip, custom designed by Yuki Watanabe of our faculty and Wayne Dettloff of MCNC [14]. The board plugs into a Sun host and can perform 580,000 inferences (flips) per second. It was delivered to Oak Ridge National Laboratory, where it is used in a robotics application. The **Blitzen** project is based on million-transistor custom chips designed at MCNC [9]. The chips,

which contain 128 bit-serial SIMD processors, can be used to configure very large X-mesh-connected arrays. Eight chips are mounted on a Sun-compatible board, designed and built by Raj Singh of our laboratory. The board has been delivered to NASA-Goddard, where it is being used to develop parallel image-processing applications for possible future, flyable systems.

2 Pixel-Planes Project

Pixel-Planes is the most ambitious project undertaken so far by the MSL. Since it has driven development of much of our environment, technology, and tools, its history serves to show how we built our capabilities over the past 10 years.

Some background: 3D computer graphics has been described as a computational task with embarrassing amounts of parallelism. Since the late 70's various efforts have been made to capitalize on *image* parallelism by computing multiple pixels in parallel for each primitive object (*e.g.*, polygon) on the screen, in order to generate 3D computer images at interactive rates. In nearly all cases, image-parallel graphics machines have a *frame buffer*, a memory that stores various indices (color, depth, *etc.*) for each pixel, which is accessed in pixel-parallel fashion by image-generating processors. Polygon visibility and shading computations are quite simple; they need only *interpolation* across pixels, and *modification* of frame buffer memory contents according to the interpolated values. The main problem with doing these computations in parallel has been connecting multiple processors to the frame buffer with sufficient bandwidth. (For example, Gouraud-shading 1M 100-pixel triangles per second, the performance level of the fastest current systems, requires at least 300M memory operations per second.)

In late 1980, while organizing and teaching UNC-CS's first course in VLSI design, Henry Fuchs conceived the basic ideas for the Pixel-Planes systems we have been building for the past ten years. He noted that bi-linear interpolation was a simple, unifying mathematical representation for many graphics computations, and that a very compact structure, a binary tree of multiply-accumulator stages, could generate interpolants simultaneously for all pixels on the screen. The remaining work of modifying the frame buffer contents was very simple and could be done by a trivial processor connected to each pixel's memory. Fuchs speculated that if both the bi-linear interpolator and the pixel processor were implemented in bit-serial form, the circuitry would be so compact that it could economically be put on the same chip with the frame buffer memory elements, thereby removing the frame-buffer memory bottleneck [6]. Upon hearing these ideas described during a class lecture, I volunteered to design a chip to try them out. This project was the beginning of a 10-year effort to exploit these simple ideas both in hardware systems and in graphics algorithms and applications. Figure 2 shows how various areas of the project developed.

Project History

Pixel-Planes 1 was designed during the pre-history of the MSL with minimal tools. The chips contained just four pixels, each with 16 bits of (shift-register) memory. Fabrication was completed during Summer 1981, and to our amazement, some of the

circuitry actually worked. But by then we had figured out how to design chips using standard, dense memory organizations. More importantly, the MSL had been formed, so there was hope of building an actual working display based on the idea.

	← Proofs-of-concept & development prototype →			← Full-scale systems →	
	Pxpl 1	**Pxpl 2 &**	**Pxpl 4.1**	**Pxpl 4.2**	**Pxpl 5**
Team	1	1	4	5(Hardware) 5(Software)	6(Hardware) 10(Software)
Tools	Minimal (CIF text entry; swith-level sim; batch DRC	none (chips designed at Xerox PARC)	°CAESAR & related (chips) °CAESAR & local ext's (boards) °Hand analysis of PC board circuits °Mac aids for schem's, mechanicals		°MAGIC & rel'd (chips) °OrCAD schem. entry °PranceGT autorouting °Mech drafting CAD
Document-ation	hand-drawn schematics, no user doc's	hand-drawn schematics, minimal user doc's	Mac-aided hand-drawn schem's and mechanicals; fairly comprehensive user manulas		Extensive, formal documentation pkg. Extensive user manuals
Computing Environ.	VAX 780 time-shared on text terminals	Time-shared VAX 780; workstation hardware host	Multiple VAXen and compute servers on extended network; workstation hardware host		Multiple workstations and compute servers; workstation h/w host
Clock	1 MHz?	1-5 MHz	8 MHz	10MHz (chips) 20MHz (video)	40 MHz (chips & bd's) 160MHz (ring network)
Chip Technology	5µ nMOS 2K transistors	4µ nMOS 5K T's (Pxpl2) 10K T's (Pxpl3) 4-20 customs total	4µ nMOS 25K T's (EMC) 10K T's (serializer) ~100 customs total	3µ nMOS 50K T's (EMC) 10K T's (serializer) 2000 customs	1.6µ CMOS: 400K T's (EMC) 125K T's (IGC) 2µCMOS, 25K T's (CT) up to 2000 customs
Board Technology	None	3-4 boards, 3M insulation displacement technology	5 wire-wrap bd's (1) custom PCB, 15x15", 6 layer, 100 components	5 wire-wrap bd's (1) 16x24" 4-layer PC w/ press-fit conn's;(32) 15x15" 6- layer PCB's, ~150 components	12 board types; most controlled impedance, 6-10 layers, 8-mil features, surface mount technology, ~50 boards ~15K comp's in system
Off-shelf Technology	None	TTL glue, video timing, A/D chips	TTL glue, PAL's, fast SRAM, Video hybrids		Many PLD's, DRAM, VRAM, 40MHz RISC proc's, 100K ECL, 200MHz Video, 160MHz RF comp's
System Size	None	3-4 boards in open card cage	Standard enclosure with 1 card cage	Large custom enclosures; multiple custom card cages, power supplies, cooling	
Laboratory Facilities	None	Scopes, logic analyser, bench instruments	Laboratory comp. with frame buffers for CADgraphics	Automated chip tester, wafer prober, electrical, thermal instr'n	CAD workstations; high speed scopes & probes; network analyser, SMT rework, new building w/ 8000 sq ft h/w lab

Figure 2. Pixel-Planes project chronology.

We were extremely fortunate early in the project to have received help from Lynn Conway's group at Xerox PARC, particularly Alan Paeth and Alan Bell. Paeth first designed a chip based on our new ideas, and the chip was fabricated by Xerox. It contained 64 pixels, each with 16 bits of 3T RAM. We built a 4-chip system, **Pixel-Planes 2**, at UNC on four small boards, with an LSI-11 providing control and data inputs. It produced 4x64-pixel images on a television monitor and took around a minute to paint three polygons. This was the most complex image that could easily be recognized, and it required a small cardboard model by way of explanation. Nevertheless

the system was a success; few university groups at that time (early 1982) had demonstrated any kind of system built from their custom chips.

Paeth also designed the core of the **Pixel-Planes 3** chips, with the rest of the circuitry added at UNC using our newly installed Berkeley tools. John Eyles, a new hire in the MSL, took on the job of designing the Pixel-Planes 3 system, which had 20 or so custom chips. We were able to demonstrate a 64x64-pixel display in 1983 that could update a 6-polygon image about once per second.

By 1984, we had funding in hand to try to develop our ideas more fully. It seemed that we had paid remarkably little attention to system-level issues during the designs of Pxpl2 and Pxpl3, so this time we set out to do things better. Within six months we designed and received back working silicon for a new memory part and for a second chip, part of the controller for the SIMD engine. **Pixel-Planes 4.1** had 32 enhanced memory chips on a board, a controller board containing the second custom, and reasonably high-quality video generation on a third board. It was shown at the 1985 VLSI conference in Chapel Hill [11], updating simple images at a few frames per second. More significantly, it demonstrated several new algorithms, invented by graduate students and others outside our group, that we had not envisioned when the machine was designed.

We presented our research results at SIGGRAPH '85 [7], with vague plans to design better chips and build another prototype. There was much interest in the paper, but the reaction, particularly to estimates of performance, made clear that many viewed our approach (correctly) as a paper tiger. A caucus of group members at the conference produced an instant and unanimous decision to build a full-scale system with the custom chips already on hand and to demonstrate it at the next SIGGRAPH.

This was a pivotal moment in the history of our research group and of the MSL, because it was the point at which we first decided to build and field a substantial, *useful* system, with no apologies made for performance or capability. The work that followed differed both in scope and quality from our previous system-building efforts. First, our chip designs had to be re-evaluated against much higher standards, since they would be made in large quantities and would populate a system that would actually be *used*. Second, we had to worry about all kinds of logistical problems with quantity fabrication and full-system assembly, including building a chip tester (a mini-project in itself), designing power and cooling systems, cabinet mechanicals, and so forth. Finally, where our earlier prototypes had addressed only *rendering* (the screen-oriented computations accelerated in the enhanced memory chips), **Pixel-Planes 4.2** was a *complete* graphics system, including a front-end geometric transformation engine, a host, and system and application software. The project nearly foundered because we underestimated the difficulty of implementing and programming the front-end processors.

We completed the machine two weeks prior to SIGGRAPH '86, drove it in a rented van to the conference site in Dallas, Texas, and successfully demonstrated the machine for a week. Its performance was nearly a factor of two higher than the fastest commercial

machines of the time, even several that were physically nearly as large. We later demonstrated the system via satellite closed-circuit television at the 1987 Stanford VLSI Conference [12]. The machine proved to be quite reliable, despite its size and complexity, and it has been in daily 24-hour use in our department's graphics laboratory. Easy to program, and with *delivered* performance (~35,000 shaded triangles/sec) still comparable to high-end systems, it has become the workhorse of our graphics research program [5].

For the past four years, our team has been developing a new machine whose goals include: very high performance, well in excess of 1M shaded triangles per second; a wide range of applications, including support for advanced lighting models and textures, volume rendering, and curved surfaces; support for graphics standards, so that the machine would be relatively easy to use; modularity, so that performance could be scaled linearly with hardware cost over a wide range. To achieve performance and scalability goals required breaking some new ground in graphics architectures [8]. **Pixel-Planes 5** is a heterogeneous multicomputer with two kinds of processors: general-purpose, math-oriented computing nodes based on Intel i860's, and small (128x128-pixel) SIMD computing surfaces, called "Renderers", that are essentially one-board versions of Pxpl4. It has a single, general, high performance interconnection between nodes, a 5Gb/sec multi-token ring network, that carries pixels, primitives, instructions, and control. The machine is the first, to our knowledge, to traverse graphics display lists in parallel. It is a hybrid parallel graphics architectures; it processes multiple streams of primitive objects in parallel *and* paints pixels within the primitives in parallel. Hardware was completed in July 1990 and was running code developed on a porting base within two weeks. Work continues to improve performance and to build new applications.

Team

Pxpl2 and Pxpl3 were each designed and built essentially by a single person. Pxpl4.1 was our first team effort. The division of labor had custom chips designed by Eyles (who also designed the controller for the SIMD array) and Poulton, (also responsible for mechanicals, power, and cooling). John Austin, who joined us in 1983, designed video and host-interface boards, developed board tools, and oversaw documentation.

The much more intensive development effort of our first full system, Pxpl4.2, required that a somewhat larger team, five engineers and a software systems guru, work together in a different way. In so large a system, many pieces had to fit together with little margin for error. We began to take documentation seriously and set higher standards for our design work. The staff had roughly the same division of labor, with two additions: Wayne Dettloff of MCNC spent about 4 months helping us with chip probing and testing. Thomas Greer, who joined the hardware team halfway through the year's effort, deserves much of the credit for successfully completing the hardware and software for the front-end processor. We also relied heavily on the MSL's 2-man facilities staff. Mark Monger contributed device drivers, system and tester software. John Thomas took on the difficult job of acquiring the large numbers of electronic parts consumed in the project.

The hardware staff for Pixel-Planes 5 grew only modestly from the Pxpl4 effort. Laura Weaver replaced Austin as video and host-interface guru. Two graduate students joined the hardware team: Steven Molnar designed the i860-based processor, and Michael Bajura did much of the logic simulation for processors and ring network. As before, we depended heavily on the lab's staff, including our secretary, Sharon Walters, who helped design several of the PC boards, and Brad Bennett, who kept our computing environment running. The Pxpl5 project differed from other MSL projects in requiring a significant software design effort, including a multicomputer operating system, a real-time hardware resource manager, and graphics library support. The software team for Pxpl5, as in previous generations, was led by Henry Fuchs and included: Andrew Bell, David Ellsworth, Howard Good, Victoria Interrante, Jonathan Leech, Carl Mueller, Ulrich Neumann, Marc Olano, John Rhoades, Andre State, Brice Tebbs, Russ Tuck, Greg Turk.

Tools

The early projects used no design automation (at UNC). For Pxpl4.1 and 4.2, we used the Berkeley tool suite, including CAESAR and related layout analysis tools. These, and the MAGIC tools that followed, deserve much of the credit for the first silicon success we achieved throughout this project (in six years we have had to re-spin only one chip). The Pxpl5 chips required large amounts of circuit simulation that would have been impossible without CAzM, a tool built jointly by Duke CS and MCNC. The chips were hand crafted much as our earlier designs, but *much* more work went in up front on simulation and analysis.

Board design capture up through Pxpl4.2 was done manually; pencil and paper were replaced by MacIntosh drawing tools when they became available. The PC boards for Pxpl4 were all done using CAESAR, augmented with some home-grown tools. For Pxpl5, board CAD went through a major revolution. We set goals that could only be achieved by automated layout, and we were fortunate to put into place a suite of tools that could do the job (though painfully!). For the mechanical designs, hand drawings were replaced by MacIntosh tools, and more recently by automated drafting tools.

Documentation

Most engineering effort actually goes into producing documents, not hardware, especially for projects that require many people. When we began Pxpl4, our first team effort, we made considerable effort to set standards for documentation and to try to stick to them. These efforts were motivated by bad experiences: parts of the project that were poorly documented didn't work and couldn't be fixed. Good documentation habits were forced on us by increasing complexity, a need to communicate accurately among ourselves and, increasingly, to others outside our group. Pxpl5 was perhaps an order of magnitude more complex than Pxpl4. The project generated many hundreds of pages of documents, controlled and organized in a fairly formal way. Specs were read, discussed, marked up, and rewritten many times. Unfortunately, documentation is still a separate activity from design capture, because we yet lack suitable tools for integrating the two activities.

Computing Environment

Our laboratory has passed through the same changes in computing environment as most academic CS departments: time-shared minicomputer; multiple networked mini-computers; multiple networked personal workstations. It seems amazing in retrospect that all of our lab's business between 1984-86 was done on a few MacIntosh's and a VAX 750 with text terminals and a couple of color frame buffers.

Technology

As shown in Figure 2, improvements in IC technology allowed us to build ever better chips. Each generation increased memory per pixel (by a little over 2x), number of pixel processors (2x), and speed.

Printed circuit design is more difficult to master than one might at first assume, for so mature a technology. Design rules are fairly complex compared to chips, and one must pay explicit attention to the manufacturing and assembly processes, mechanical compatibility, thermal constraints, and signal integrity issues. We were able to produce boards for Pxpl4 using relatively crude tools, boards that almost functioned correctly on the first spin, because clock speeds were relatively low. The few changes that were required arose from poor understanding of transmission line behavior. In Pxpl5, many boards have TTL/CMOS circuitry running at 40MHz, and the ring network design, implemented in 100K ECL, runs at 160MHz. We took the lessons of Pxpl4.2 to heart and spent much effort on the details of signal integrity and transmission-line characteristics. We simulated the logic of all the designs fairly thoroughly, no small task given that we had to generate models for processors, custom chips, DRAM's, VRAM's, and PLD's. These efforts were worthwhile; the boards worked on first spin with very few patches, and critical signals were very clean.

We undertook one other technology adventure in this project. The ring network design is greatly simplified by having a globally synchronous 160MHz clock, which we distribute with low skew over multiple card cages using a method suggested by Vernon Chi called "salphasic" [3]. In this approach, the clock is transmitted as a sinusoidal signal on reactively terminated transmission lines, thus setting up a standing wave that is, by definition, in phase along the line. This method turned out to work very well, achieving clock skews of a few hundred picoseconds across a pair of system backplanes.

System Size

Our prototypes progressed from simple card cages to small standard enclosures to full systems in racks with custom card cages, power and cooling systems. These larger systems required surprising amounts of effort to design mechanical parts, DC distribution, AC power control, and so forth, even though we used only very simple, conservative packaging techniques, with standard-size cards and cages and off-the-shelf components wherever possible. Others who are attempting to build large experimental systems that require advanced packaging might be well advised to proceed incrementally.

Lab Facilities

System design projects in the MSL cycle between design and implementation phases and therefore tend to force the development first of CAD facilities, then fabrication and test. One of the many benefits of a shared laboratory is that other projects are also pushing laboratory development. Frequently, others have paved the way in some new technique or tool; about equally frequently we have returned the favor. Pxpl5 needed a fleet of design workstations, whose compute cycles, main memory, and disc space were pushed hard by this complex project. Its high-speed designs led to a heavier investment in electrical measurements instrumentation. A network analyser purchased during the project was invaluable for developing insight about transmission lines and for producing empirical circuit simulation models. We also advanced the lab's packaging facilities, using surface mount assembly extensively on most of the system's PC boards. The only major costs were the purchase of a good rework station and the time needed to learn the design issues.

In the near future, increasing complexity in the software systems that run on our experimental machines will require laboratory staff whose main job is software development and management. In hardware design, we are looking for ways to speed up chip designs, freeing up our scarce human resources for more productive activity than pushing rectangles, and at alternate methodologies, particularly asynchronous design. We will certainly be designing high-speed chips and systems, so instrumentation and design tools for analyzing high-speed signals and components will be essential.

3 Conclusions and Observations

What worked

The shared facility concept of the MSL has exceeded expectations. The lab's collegial environment has made the work fun, staff have stayed in the organization for extended periods, and so knowledge and skills have been built and retained. The equipment and facilities are superior to any that a single hardware project could reasonably afford. With accumulated know-how and superior equipment, we have been able to launch new projects quickly and inexpensively, to carry the ideas through to full-scale system prototypes, and to field these machines, maintaining them in operation over long periods.

The non-hierarchical organization of the lab has worked extremely well. Since there's no-one else to do the grunt labor, staff engineers have ended up doing everything from architecture to soldering. I believe this has been useful: knowing one will have to deal with low-level details, one pays careful attention to getting them right. Several students have had rewarding experiences in the laboratory. Staff members have acted as mentors and as knowledge resources, though in some cases these roles have been reversed, with the student acting as technical leader and the staff member as assistant.

The productivity of our staff has grown with increasingly powerful tools, much as we predicted ten years ago. The number and complexity of systems we have built and

delivered has risen dramatically, especially in the last five years, while the staff has grown only from 4 to 10.

We have been successful at judging appropriate levels of risk to take for the projects we have built. None has failed because we didn't master some hardware technology; on the other hand, we have built some fairly aggressive systems.

What didn't work

The MSL made two unsuccessful attempts to build advanced technology in-house. In the first instance, we imported a new technique for building ceramic-based thick-film hybrids, based on directly writing conductors and insulators on a substrate. It offered the possibility of producing dense first- and second-level packages that could be prototyped very rapidly and seemed a perfect match to our circumstances. We believed that having the technology in-house might allow us to attempt package-driven prototypes that would otherwise be impossible. The project never achieved its promise. First, full-time commitment was needed to really make this technique work, but our small staff was already busy trying to keep up with our main business, building experimental systems. Second, there was no "customer", a system project that depended critically on the technology. Third, the window for the technology was even more limited than we had imagined (for example, current PC boards can readily achieve the circuit densities the technique was capable of), so without ongoing improvements, it rapidly fell behind.

In the second instance, we invested in an E-beam voltage contrast probe for directly measuring on-chip signals in real time. This effort was motivated by early experiences with chip designs that had serious circuit-level problems, difficult to analyse from outside the chip. This project also languished from having insufficient staff, no demanding customer, and technology that moved ahead much faster than expected, rapidly obsoleting our equipment. Trends in chip design also made the approach less useful than we had expected. Our own chips tended to be designed conservatively, and for these, circuit simulation turned out to be reliable and accurate. No exotic testing was really needed, because the chips tended to work on the first crack.

These projects didn't pay off perhaps because the relatively small scale of our lab. We have a local model for a shared facility on a much larger scale in the Microelectronics Center of North Carolina, which, interestingly, has developed a very active advanced packaging research group and a first-class E-beam probe research team.

What we could do better

We have been only modestly successful at involving students in system building. We've provided an intensive learning experience, but only for the handful of students who were involved in hardware projects. Currently we are thinking about a curriculum that could provide a more structured experience for more students, with the MSL serving as the major resource for laboratory course work. This experiment will not be without risk. An education role for the MSL will inevitably divert resources from research, but the issue is so important that we feel we must try.

We have not invested heavily enough in design automation, particularly for silicon design. If the productivity trend we have enjoyed is to continue, we'll have to do better.

Little technology has been transferred out of the laboratory. Currently we are working to change the situation, trying two different models. One is an intense, focussed, joint development project with a small startup, the other a longer term joint research effort with a larger company.

The foregoing is an entirely personal, nuts-and-bolts view of experimental system building at our university. Reference [2] is a more reflective account of the motivation and philosophy of the MSL. The Pixel-Planes project is supported by the National Science Foundation (MIP-9000894) and the Defense Advanced Research Projects Agency, Information Science and Technology Office (Order No. 7510).

References

[1] Bishop, G., and Fuchs, H., "The Self-Tracker: A Smart Optical Sensor on Silicon," *Proc. of the Conf. on Advanced Research in VLSI*, MIT Press, 1984, pp. 65-73.

[2] Chi., V. and Dollas, A., "Rapid System Prototyping in Academic Laboratories of the 1990's," *Workbook of the First IEEE International Workshop on Rapid System Prototyping*, June 4-7, 1990, Research Triangle Park, NC.

[3] Chi., V., "Salphasic Distribution of Clock Signals," UNC-CS Tech Report 90-026.

[4] Ericksen, J., Pizer, S., and Austin, J., "MAHEM: A Multiprocessor Engine for Fast Contrast-Limited Adaptive Histogram Equalization," *Proc. of SPIE Medical Imaging IV: Image Processing*, Feb. 6-8, 1990, pp. 322-333.

[5] Eyles, J., Austin, H., Fuchs, H., Greer, T., and Poulton, J., "PIXEL-PLANES 4: A Summary," (presented at Eurographics '87, 2nd Workshop on Graphics Hardware), *Advances in Graphics Hardware II*, Springer-Verlag, 1988, pp 183-207.

[6] Fuchs, H., and Poulton, J., "Pixel-Planes: A VLSI-Oriented Design for a Raster Graphics Engine," *VLSI Design*, Vol. 2, No. 3, pp 20-28, Q3-1981.

[7] Fuchs, H., Goldfeather, J., Hultquist, J., Spach, S., Austin, J., Brooks, F., Eyles, J., and Poulton, J., "Fast Spheres, Shadows, Textures, Transparencies, and Image Enhancements in Pixel-planes," *Computer Graphics*, Vol. 19, No. 3 (Proceedings of SIGGRAPH '85), pp. 111-120.

[8] Fuchs, H., Poulton, J., Eyles, J., Greer, T., Goldfeather, J., Ellsworth, D., Molnar, S., Turk, G., Tebbs, B., and Israel, L., "A Hetergeneous Multiprocessor Graphics System Using Processor-Enhanced Memories," *Computer Graphics*, Vol. 23, No. 3 (Proceedings of SIGGRAPH '89), pp 46-51.

[9] Heaton, R., Blevins, D., and Davis, E., "A Bit-Serial VLSI Array Processing Chip for Image Processing," *IEEE JSSC*, Vol. 25, No. 2, pp. 364-368, 1990.

[10] Kedem, G. and Ellis, J., "The Raycasting Machine Prototype," *Proc. of the 1984 IEEE International Conference. on Computer Design*, pp. 533-538.

[11] Poulton, J., Fuchs, H., Austin, J., Eyles, J., Heinecke, J., Hsieh, C-H., Goldfeather, J., Hultquist, J., and Spach, S., "PIXEL-PLANES: Building a VLSI-Based Graphic Systems," *Proc. of the 1985 Chapel Hill Conf. on Advanced Research in VLSI*, Computer Science Press, pp. 35-60.

[12] Poulton, J., Fuchs, H., Austin, J., Eyles, J., Greer, T, "Building a 512x512 Pixel-Planes System, " *Proc. of the 1987 Stanford Conf. on Advanced Research in VLSI*, MIT Press, pp. 57-71.

[13] Wagner, R.A., "The Boolean Vector Machine [BVM]," *Proc. of the 10th International Symposium on Computer Architectures*, Stockholm, Sweden, June 13-16, 1983, pp 59-66.

[14] Watanabe, H., Dettloff, W., and Yount, K., "A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture," *IEEE JSSC*, Vol. 25, No. 2, pp. 376-382, 1990.