Written by
Nick Tredennick

DynamicSilicon

Published by
Gilder Publishing, LLC

The Investor's Guide to Breakthrough Micro Devices

The Evolution of Digital Design

Why are PLD companies like Altera (ALTR) and Xilinx (XLNX) on our list while ASIC companies like LSI Logic (LSI) are not? In electronic design, application-specific integrated circuits (ASICs) connote high performance, high density, and low cost for high-volume uses. Think DVD player.

Programmable logic devices (PLDs) connote flexibility, low performance, and high cost for prototyping. Think latest Cisco (CSCO) router. Continuous improvements in PLDs raise their performance and lower their cost, increasing the uses for which PLDs and ASICs compete. ASIC vendors are adding programmable logic to their chips. The PLD vendors are adding "hard" macros. Are these businesses converging? Or are companies like Altera and LSI Logic practicing "diworsification?" Altera, a PLD company, is making chips that can't be changed, and LSI Logic, an ASIC company, is making chips that can be changed. What strategy will dominate? We'll develop the insight to answer these questions by tracing the evolution of digital design from the introduction of integrated circuit building blocks—"IC macros."

Each technology wave changes the designer's job. As Moore's law drives more transistors onto the chip, the design process changes. The productivity and the skills of the designers change. The level of abstraction for the engineer changes. By level of abstraction, I mean the units that are the building blocks for the translation of the written specification into the physical hardware. In the early days of digital design, the level of abstraction was the inverter and the AND gate. Working with inverters and AND gates was simpler and quicker than designing with individual transistors, resistors, capacitors, and other "discrete" components. As I trace the evolution of digital design, I'll point out changes in the design process, designer productivity, engineering skills, level of abstraction, verification and debug methods, and changes in the flexibility and capability of the end systems.

CEOs vote themselves big compensation packages for this kind of strategic insight: "We need a system that'll knock the socks off the competition!" This "mission statement" goes to "system architects" who write the specification for the system. System architects propose a block diagram like the one in fig. 1 and they write an English-language description of what it does. The specification includes the instruction set for the central processing unit (CPU).

The plain-English description of what the CPU does is called the "programmer's reference manual." The 116-page manual for Altera's Nios embedded processor, for example, lists each of the processor's instructions, gives its representation as ones and zeroes, and tells the programmer what the instruction does in English. The instruction labeled "SUB," for example, describes the process that "subtracts the contents of [register] RB from [register] RA, [and] stores the result in RA."

The logic design engineer translates the English description of instruction behavior into the data paths and control circuits that enable the Nios CPU to recognize "000010 00101 00100" as an instruction to subtract register R4 from

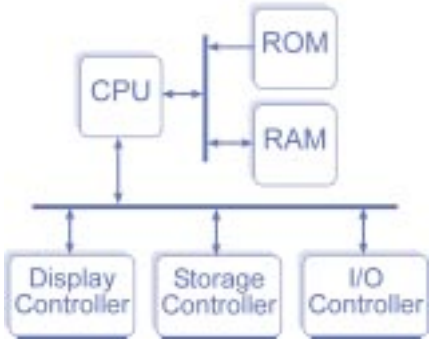
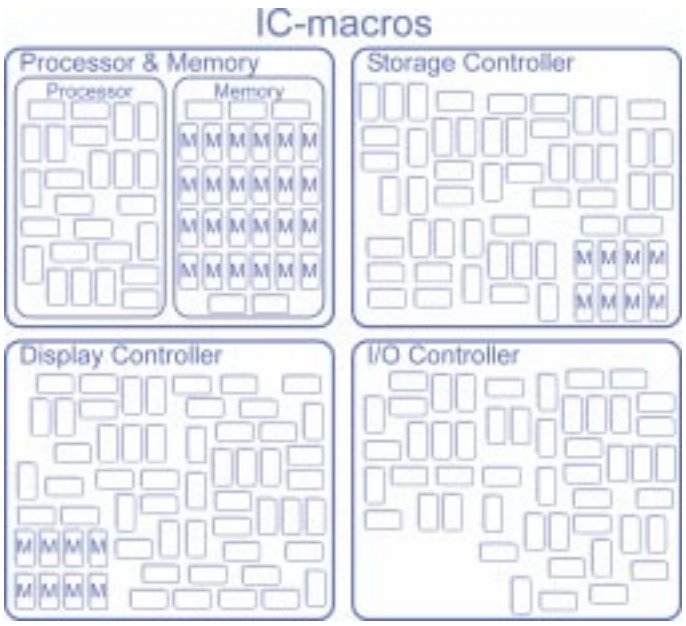


Fig. 1. A system block diagram is the starting point for a design.

Vol. 1, No. 10
October 2001



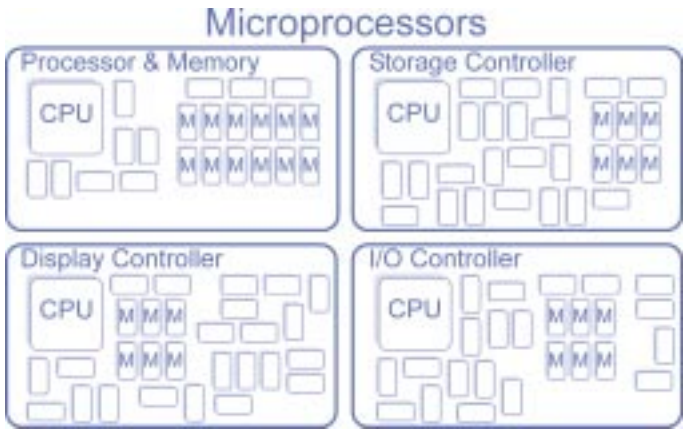


Fig. 2. (left) The integrated circuit enabled construction of complex systems. The small rectangles inside each subsystem are IC macros.

Fig. 3. (above) The microprocessor brought the computer's programming model to embedded systems and consolidated IC macros.

register R5 and to put the result in R5. *The logic design engineer bridges the gap between the system's functional description and its physical realization.*

Integrated circuit macros. Engineers built systems for the surging minicomputer market of the '60s and '70s with IC macros. Fig. 2 shows the result of mapping the diagram of fig. 1 into an IC macro implementation. This was the pre-microprocessor era, so the CPU was built of IC macros and occupied its own board. Its random access memory (RAM, labeled "M" in the figure) occupied another board. The storage controller, display controller, and I/O (input/output) controller were all custom designs. Like the CPU, each occupied its own board and was its own unique design.

IC macros changed the designer's job. The logic design engineer converted the English-language description of the system's behavior (including the CPU's description) into IC macros. Productivity improved dramatically because the engineer did not have to worry about individual circuits and transistors. The IC macros connected to each other in standard ways. Each IC macro can be hundreds or thousands of transistors. Nothing was supplied except the collection of IC macros to be used as the building blocks. The engineer had to construct the registers, buses, arithmetic units, and controllers. There was no system simulation; the engineer debugged with an oscilloscope. It took redesign to correct errors. The IC macros did not need either circuit or logic simulation, since they were commercially available blocks that were previously verified. The end system was inflexible.

Microprocessors. Intel introduced the first commercial microprocessor—CPU on a chip—in 1971, but it wasn't until the mid to late '70s that microprocessors invaded computer systems designs. The microprocessor-based workstation market of the '80s wiped out the IC macro-based minicomputers. The minicomputers, with their custom and proprietary CPU designs, gave way to workstation designs based on commercially available microprocessors. Sun Microsystems built a great business based on this strategy.

Moore's law shrunk the system to fig. 3 and it made the system more capable; the microprocessor changed the designer's job. The CPU (microprocessor) provided

DynamicSilicon

Editors	**Nick Tredennick**
	Brion Shimamoto
Publisher	**Lauryn Franzoni**
Web Editor	**Jorin Hawley**
Designer	**Julie Ward**
Subscription Services	**Melissa McNally**
Executive Editor	**Richard Vigilante**
President	**Mark T. Ziebarth**
Chairman	**George Gilder**

Dynamic Silicon is published monthly by Gilder Publishing, LLC. Editorial and business address: 291A Main Street, Great Barrington, MA 01230. Editorial inquiries can be sent to: bozo@gilder.com.

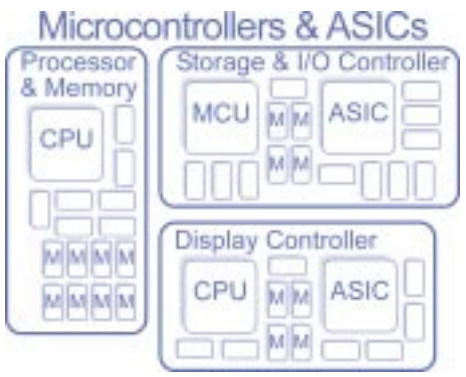


Fig. 4. ASICs and microcontrollers consolidated board-level functions onto the chip.

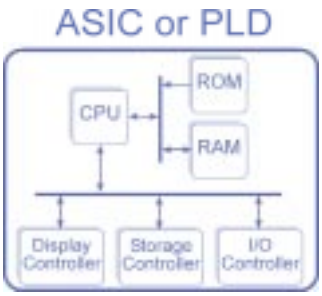


Fig. 5. ASICs and PLDs consolidated all functions in the original block diagram.

the controller for each subsystem design. The engineer no longer built a custom controller for the storage subsystem, for example. The design process changed from one of mapping a specification into IC macros to one of *programming* the specified behavior on an embedded CPU. The subsystems still used IC macros to build data paths, to build interfaces to other subsystems, and to provide logic-gate odds and ends (called "glue logic") that tied the system together.

Designing subsystems with CPUs brought the computer's programming model to embedded systems. This changed the design process in two ways. First, a substantial part of the design process became programming. Second, the engineer no longer designed the control structure of the system—it came as the CPU. The engineer built the necessary data paths and programmed the subsystem behavior. Moore's law and the CPU acted to consolidate IC macros thereby shrinking the system.

Shifting to a programming model has significant consequences. First, there are more programmers than controller designers, opening subsystems to more uses. Second, programming tools developed in computer environments apply in this domain, extracting value from investment elsewhere and raising the level of abstraction (making the problem-solving engineers more productive). Third, the programming environment permits simulation and debugging at a level that's more efficient than using an oscilloscope. Engineers are able to write and debug software on general-purpose computer systems before the subsystem hardware is built.

The system is flexible. Programming revisions can correct errors and allowed some migration of function for fielded systems.

Microprocessors displaced IC macros in designs because microprocessors simplified the engineer's job by providing the controller and by bringing programmed solutions to the embedded environment. Microprocessors do not displace IC macros because they are faster. They aren't. IC macro-based designs can be faster because they implement functions more directly. If performance is primary, the microprocessor loses. Most electronic applications, however, are cost sensitive; they are not primarily concerned with performance. Since the microprocessor is programmed, the design can be cheaper; since there are fewer chips, it can be cheaper to build.

Microcontrollers and ASICs. IC macros can't survive Moore's law. Once a chip becomes pad limited, it doesn't get smaller, faster, or cheaper, so it can't compete with chips that do (*Dynamic Silicon*, Vol. I, No. 3). Commonly implemented functions (such as serial and parallel interfaces, transmitters and receivers, counters, and timers) migrate onto the same chip as the microprocessor. The microprocessor then becomes a "microcontroller" (MCU in fig. 4).

Performance-oriented functions migrate to ASICs, displacing both IC macros and CPU program code. Functions in the CPUs and in the MCUs retain the advantages of programming for development and for field revisions. ASICs add performance, reduce the part count, and lower cost, but they are inflexible.

Engineers design, simulate, and debug functions that map into those CPUs and MCUs on separate general-purpose computers.

Engineers also design, simulate, and debug ASICs on general-purpose computers. But Moore's law has driven the capacity of these chips to millions of gates. The effort required to design, simulate, and debug increases at the same rate as the design complexity. For example, engineers care about three levels of simulation: function, logic, and circuit.

Simulating circuits, at a "black box" level, on a computer is feasible. Simulating circuits at their ones and zeros (logical) level is now borderline feasible. Circuits have become so complicated that simulating their logical operation takes huge amounts of computing power. It is better to recreate the logic of circuit using PLDs. The PLD-based simulation runs thousands of times faster than a computer-based one. Simulating large circuits' electrical behavior

isn't practical. This is an ongoing problem for ASICs. ASICs are large custom circuits. PLDs consist of standardized electrical components that are much easier to verify.

System on a chip. By now the pattern is clear: Moore's law drives component consolidation. More stuff going on in one place. Consolidation changes the design methods and it simplifies the skills required of designers (increasing the pool of eligible designers as the level of abstraction rises). Consolidation changes design verification and debugging procedures and it changes the capability and the flexibility of the resulting system.

Microprocessors consolidated IC macros. Microcontrollers and ASICs further consolidated IC macros. In the current phase (fig. 5), logic chips swallow the subsystems of the system block diagram. IC macros, CPUs, MCUs, and the functions from last generation's ASICs all fit on a single chip.

If you are a CPU or MCU vendor, such as Triscend or Cypress Microsystems (CY), your point of view is that the CPU or MCU is sucking in all of the ASICs and IC macros around it. If you are a PLD or ASIC vendor, such as Altera or LSI Logic, your point of view is that the PLD or ASIC is sucking in the CPU and IC macros around it. The point of view has bearing on market positioning and on the application space for products, but the groups face similar transitions. In microprocessor and ASIC implementations, where the CPU and the ASIC are independent chips, the value of the CPU or the ASIC is in the silicon. In the system-on-a-chip implementation, the value has migrated from the silicon to the design database for each function block (e.g., fig. 5's display controller, storage controller, or CPU).

The designer's job evolves from building interfaces among chips on a board to building interfaces among databases of functions that will be implemented on a single chip.

For microcontroller and ASIC designs and for system-on-a-chip designs, the PLD offers an alternative to the ASIC. As tradition has it, the engineer builds a PLD-based prototype for initial test and debug, and perhaps for initial production, and then converts the PLD to an ASIC for high-volume production. If only a few units will be built, it may be cheaper to build PLD-based systems and to skip the ASIC design. Where's the cost crossover between low-rate production with PLD-based systems and high-volume production with ASICs?

PLDs and ASICs

Fixed and variable costs. Suppose you are building a system and must choose between an ASIC-based implementation and a PLD-based implementation. Fig. 6 shows the tradeoff. The vertical axis is the total component cost of ASICs or PLDs to build a number of systems. The horizontal axis is the number of systems built. The steep line beginning at the origin is the cost to implement the system with PLDs. The modestly sloping line beginning at "fixed costs" is the cost to implement the system with ASICs.

Your engineers design the system by mapping the system's functions into logic that will be implemented either in a PLD or in an ASIC. If the system is PLD-based, the engineers buy standard PLDs from the manufacturer or from the distributor. Since the manufacturer amortizes the cost of designing the chip, the customer sees only a unit price. If the system is ASIC-based, the engineers produce a design file and your company contracts with an ASIC foundry to make the IC masks and to build the chips. The costs that are independent of the number of chips the foundry builds are called non-recurring engineering (NRE). The design work for either implementation is about even. For the PLD implementation, there is a high component cost and no NRE. For the ASIC implementation, there are fixed costs, but the chips are individually cheaper than PLDs. For fig. 6, I assumed a PLD to be six times the cost of an ASIC.

If you build only tens or hundreds of systems, the PLD implementation will be cheaper (the shaded area in fig. 6) because the ASIC's NRE will be spread over only a few systems. Suppose the ASIC's NRE is $200,000 and the chips are $4. An equivalent-capacity PLD might be $24. If you build only 1,000 systems, the NRE contributes $200 to the cost of each chip, so the ASIC implementation is more expensive than the PLD implementation. If you build 100,000 systems, the NRE contributes only $2 to the cost of each chip, making the ASIC cheaper. To keep the analysis simple, I assume no volume discounts. I also assume that the engineer's ASIC design file builds a working chip on the first try. (We'll make adjustments later.) Fig. 6 seems a reasonable analysis, but it has a shortcoming: it's static.

PLDs, ASICs, and time

You won't buy all the components and build all the systems in a week. It's likely that you will buy components incrementally and build systems over two years or so. Buying PLDs incrementally over two years is easy; in fact, since component costs drop about 60% per year, you'll buy PLDs just in time to build the systems to take advantage of the latest price reductions. For the ASIC implementation, the NRE is an up-front cost and the lifetime supply of ASICs may be contracted at the project's start. The ASIC's per-chip cost won't be decreasing at 60% per year. Your

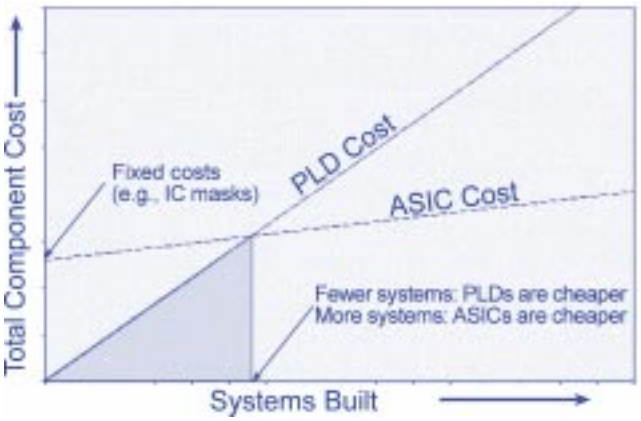


Fig. 6. Build a few systems and PLDs are cheaper; build many and ASICs are cheaper.

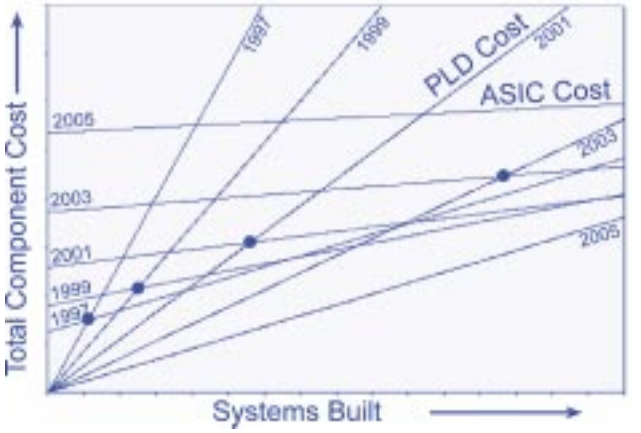


Fig. 7. Per-chip cost of fixed function decreases over time. ASIC NRE rises with semiconductor process complexity. The equal-cost point moves rapidly in favor of PLD-based implementations. (Not to scale; rates compressed for illustration.)

foundry won't want to dribble your ASIC through its line over your product's market life. The foundry will want to crank up the line and deliver all your chips in a single run so that it can reconfigure its production line for the next customer's product. Suppose you buy enough chips for a sales forecast of 25,000 systems. If you sell only 5,000, then you throw out 20,000 ASICs you paid for. If you sell 50,000, you won't be getting the second 25,000 chips at a discount. It's more likely to be at a premium—if the foundry can do it at all (if it's been two years, the foundry's current processes may be incompatible with your design).

Moore's law drives the semiconductor business by decreasing the cost for a fixed function (your design) by a factor of two every eighteen months. Fig. 7 shows the effect of Moore's law on the equal-cost point between ASIC-based and PLD-based implementations. Lines radiating from the origin in fig. 7 represent the cost of PLDs over time. The steepest line is 1997. By 1999, a PLD with the same capacity was 36% of the 1997 price; in 2001, it was 36% of the 1999 price, and so on. This figure is not to scale. The PLD cost line is falling over at 60% per year; I compressed it to 20% to illustrate a rapid trend without having to resort to (confusing) logarithmic scales. Horizontal lines cutting across fig. 7 represent the cost of ASICs. At the bottom is 1997. The 1997 ASIC line has a slope that reflects a per-chip cost that is a sixth of the 1997 PLD's per-chip cost. The slopes get flatter in successive years as per-chip costs drop. Costs start higher each year because the NRE rises with process complexity.

The equal-cost point between ASICs and PLDs is the intersection of the ASIC cost line with the PLD cost line for each year. The point isn't just moving to the right in favor of PLDs, it's *accelerating* to the right. It accelerates because the ASIC cost line is rising *and* the PLD cost line is falling over. Each year favors PLD-based implementation for higher-volume applications. As they say in infomercials, "but wait, there's more." To this point, I've treated ASIC-based design as the equal of PLD-based design. I sort differences into technology drivers, market drivers, and engineering drivers.

Technology drivers

Moore's law, mask cost, chip cost, and intellectual property are technology drivers.

Moore's law. *Moore's law works for the PLD and it works against the ASIC*. Think of the application space for PLDs and ASICs as a distribution running from a few thousand gates to millions of gates. ASICs own the high-capacity applications; PLDs own the low-capacity end; and ASICs and PLDs compete for the (vast majority of) applications in the middle. Moore's law raises the capacity of the PLD and of the ASIC. Since the PLD is coming up from lower capacity, it is encroaching more on ASIC applications. ASIC applications move up as the scale itself expands (applications get bigger as engineering productivity improves), but the scale isn't expanding as fast as chip capacity is rising, so the PLD is taking over ASIC applications.

Mask cost. Mask costs rise as the semiconductor process becomes more complex and as process geometries shrink. Rising mask costs hurt ASICs more than they hurt PLDs. The ASIC, as its name implies, is built for a specific application, so the ASIC's mask costs amortize across the volume for one application. The PLD, by contrast, is a general-purpose component, so its mask costs amortize across the volume for its entire range of applications.

Intellectual Property. ASICs implement "hard" core IP. Hard cores are physical circuits designed for a

specific semiconductor process. PLDs (generally) implement "soft" core IP. Soft cores are logic that is designed to download into a PLD. *Soft-core IP doesn't care whether the PLD is implemented in a 0.18-micron or a 0.13-micron semiconductor process*. For the PLD vendors, chip sales subsidize development of IP that is then portable to subsequent chip generations. This is not the case for ASIC IP, which is unique to a semiconductor process and may be unique to a customer.

A function implemented as a hard core will be smaller and faster and will use less power than its equivalent implemented as a soft core. But the soft core offers portability and it offers flexibility. Flexibility is options offered to the user to customize the macro for a particular application. A hard macro is what it is, but a soft macro may let the user select data widths, pipeline stages, algorithms, bus interfaces, and other options.

Market drivers and engineering drivers

System cost, performance, power, time to market, and system flexibility are market drivers. Engineering drivers include design ease, design cost, development tools, designer expertise, component availability, and the cost of errors.

Time to market. PLDs have the advantage in time to market. Even if the design is to be based on ASICs, there is likely to be a PLD-based prototype. If the system is PLD-based, production systems are close behind the PLD prototype.

System flexibility. If you're sure of *selling* millions of copies within a short time and you're sure there will be no design changes, the ASIC will be cheaper. If performance is paramount, the ASIC will be faster. PLD-based designs *are* the embodiment of flexibility.

Design ease. With PLD-based designs, engineers can add system features in increments. PLD-based design, rather than relying on black-box or on logic simulation, builds the functions and configures the PLDs that *are* the chips of the end system. The engineer can run the implementation for debugging. Since the function is running on programmed hardware, there's no penalty for evolving the design (as there would be in building a series of incrementally improved ASICs). The designer might implement and test basic elements of the system before adding complex features. Complex features can be added and tested one at a time. The ASIC designer might do this with a black box simulation or with a logic simulation, but cannot do so with the chip.

Component availability. In a PLD-based design, chips are available at the project's *start*. In an ASIC-based design, chips are available at the *end* of the design. System makers buy PLDs on the open market where other buyers contribute to chip production volume and, therefore, contribute to decreasing chip cost over time. If production falls short of PLD purchases, the chips go into another project's system. System makers buy ASICs in contract lots. If production falls short of purchases, the extra ASICs are thrown out. If demand exceeds component availability, the system maker negotiates with the supplier for another contract lot.

Cost of errors. In an ASIC-based system, the cost of an error is most probably a new "spin" of the chip, which costs another mask set and a few months. In a PLD-based system, the cost of an error may be as low as the cost to send a new configuration file to the system over the Internet.

ASICs and PLDs converge

High mask costs make developers reluctant to design ASICs. ASIC vendors attempt to counter this trend by offering application-specific standard products (ASSPs). An ASSP is the equivalent of an ASIC; it is an application-specific chip, but it is sold as a standard product rather than to a single customer. This amortizes the mask cost among the customers, but the disadvantage is that the customers cannot differentiate their products based on the ASSP's features (as they could with an internally designed ASIC).

Led by LSI Logic and by NEC (NIPNY), ASIC vendors are beginning to offer programmable logic blocks in ASIC designs. In an ASSP, this offers the customer a means to differentiate features in the system implementation.

By contrast, Altera and Xilinx have huge libraries of soft-core IP and they have components with hard-core macros as well.

Altera and Xilinx

Bob Hartmann, Altera's then VP of Business Development, hired me as Chief Scientist in 1993. Altera was a circuit design company, but, when its chips grew to over 10,000 logic gates, Bob could see the business changing and hired a logic designer (me) with a view from the next level of abstraction (logical blocks rather than circuits). When I left Altera a few years later, it was still a circuit-design company whose products were chips.

Recently, I visited Altera thinking that it would still be a circuit design company, so I was shocked by the change. Today Altera is a software and systems design company. Altera hasn't stopped designing chips. It still has cube farms of competent circuit designers. Today, Altera has as many programmers as it has circuit designers and it has alliances with intellectual property developers.

The presentations, product overview, design tools, intellectual property, embedded products, and its new "HardCopy" program, reflect Altera's current orientation and its strategy. "HardCopy" is a bridge between PLDs and ASICs. I'm using Altera as the example because I was just there to get the latest information, but the discussion applies to Xilinx as well. From an electronic components point of view, Altera has three types of products: product-term PLDs, general-purpose PLDs, and application-oriented PLDs. Product-term PLDs and general-purpose PLDs are the traditional business of Altera (and of Xilinx). Application-oriented PLDs are new.

Product-term PLDs. Product-term PLDs, called CPLDs, are low-cost, high-speed PLDs that consolidate IC macros. These PLDs are based on EEPROM (electrically erasable programmable read-only memory). They house the design's "glue logic." In product-term PLDs, Altera's MAX 3000 and MAX 7000 series compete with Xilinx's XC9500 and CoolRunner series.

General-purpose PLDs. General-purpose SRAM PLDs serve Altera's traditional prototyping market. For Altera, these are the APEX chips at the high end and the ACEX chips at the low end. For Xilinx, it is the Vertex chips at the high end and Spartan chips at the low end.

Application-oriented PLDs. The application-oriented PLDs are a new strategic direction for Altera (and for Xilinx). Altera's Excalibur and Mercury products are intended for the system-on-a-chip market. Altera is shipping a family of Excalibur chips with an ARM 32-bit processor core (see fig. 8). Altera has licensed the MIPS 4Kc 32-bit processor core, and it has announced products, but it is not shipping any MIPS-based chips.

The ARM is a hard core that includes peripherals (trace module, interrupt controller, universal asynchronous receiver/transmitter, general-purpose timer, and a watchdog timer). The added logic also includes both single-port and dual-port memory. The MIPS version will be identical except that the MIPS core replaces the ARM core and a test interface replaces the trace module. The processor and memory are added as a strip to the top of an EP20K100E to build the EPXA1. The EP20K400E and the EP20K1000E extend the '100E's 100,000-gate logic area to 400,000 gates and to 1,000,000 gates, respectively. These become the EPXA4 and EPXA10, respectively, by including the processor and peripherals logic and additional memory.

As shown in fig. 8, the APEX chips use scaled versions of a common layout. The Excalibur chips modify the APEX chips to add the logic and memory strip. Altera's *chip design* strategy is good, but there's a problem with the *product* strategy. A year ago Altera announced Excalibur with an ARM processor and then promised that the MIPS-based version would follow in the first half of 2001. Altera was also talking to Motorola about licensing a PowerPC core. There's still no MIPS-based Excalibur and there's no hint of a PowerPC. There probably won't be any. Xilinx has licensed the PowerPC from IBM and QuickLogic is shipping chips with a MIPS core. It isn't that Altera couldn't produce these chips or even that Xilinx and QuickLogic have introduced chips with PowerPC and MIPS cores first. *The real reason not to ship them is that it leads to parts proliferation, which works against Altera's fundamental business advantage.*

The fundamental advantage of the PLD makers is in making zillions of copies of identical chips that are *later* personalized for specific applications. As the PLD makers move into the system-on-a-chip business, there's a temptation to add hard cores to the chip to improve performance for specific market segments. But putting hard cores on the chip fragments chip production. Fragmenting chip production adds to mask costs and it spreads production volumes across more chip types. The soft core processor that was 40% of the chip and ran at 66 MHz last year will soon be 20% of the chip and will run at 133 MHz. The soft core may seem expensive now, but it scales with semiconductor process (as well as with design improvements and with design tool advances) without the need for redesign. *And that's why PLDs will succeed.*

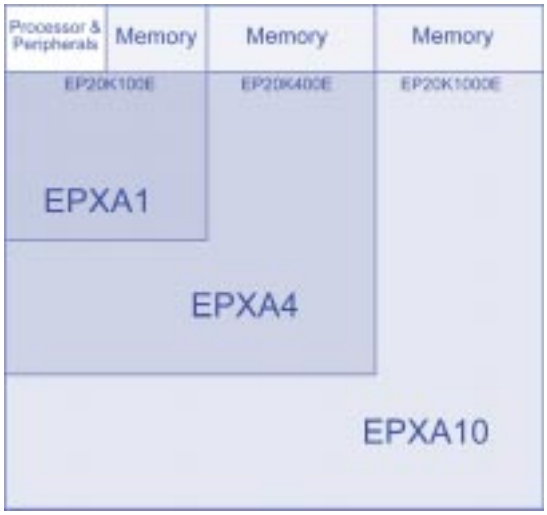


Fig. 8. Altera's Excalibur chips add a strip with a hard-core processor and peripherals and memory to members of its APEX family.

The hard-core processor and common peripherals convey the same advantages to the PLD chip that the microcontroller conveyed to board designs. The CPU brings the

computer's programming model to the chip and it provides the controller. The CPU can boot first to load configurable functions. A hard-core microcontroller (CPU and common peripherals) on the PLD encourages system-on-a-chip applications. But the parts proliferation that results from embedding a variety of hard cores moves the PLD vendors in the wrong direction.

I've traced the evolution of digital design from IC macros to the present. Technology waves change the designer's job, but the transitions aren't abrupt, so the trends are difficult to see. Today, we're moving from microcontrollers and ASICs to PLDs. *ASIC and PLD vendors are not converging despite the appearance of hard-core IP on PLDs and of programmable logic on ASICs. The PLD will continue to displace ASICs and it will soon invade the much larger markets (like your cell phone, DVD player, and PDA) held by microprocessors and by digital signal processors.*

Nick Tredennick and Brion Shimamoto
October 22, 2001

Dynamic Silicon Companies

The world will split into the tethered fibersphere (computing, access ports, data transport, and storage) and the mobile devices that collect and consume data. Dynamic logic and MEMS will emerge as important application enablers to mobile devices and to devices plugged into the power grid. We add to this list those companies whose products best position them for growth in the environment of our projections. We do not consider the financial position of the company in the market. Since dynamic logic and MEMS are just emerging, some companies on this list are startups.

Company (Symbol)	Technology Leadership	Reference Date	Reference Price	9/28/01 Price	52-Week Range	Market Cap.
Altera (ALTR)	General Programmable Logic Devices (PLDs)	12/29/00	26.31	16.38	14.66 - 51.38	6.3B
Analog Devices (ADI)	RF Analog Devices, MEMS, DSPs	12/29/00	51.19	32.70	29.00 - 93.31	11.8B
ARC Cores (ARK**)	Configurable Microprocessors	12/29/00	£3.34	£0.30	£0.25 - 3.90	£108M
Calient (none*)	Photonic Switches	3/31/01				
Celoxica (none*)	DKI Development Suite	5/31/01				
Chartered Semiconductor (CHRT)	CMOS Semiconductor Foundry	7/31/01	26.55	17.25	16.06 - 60.88	2.4B
Coventor (none*)	MEMS IP and Development Systems	7/31/01				
Cypress (CY)	MEMS Foundry, Dynamic Logic	12/29/00	19.69	14.86	13.72 - 42.56	1.9B
QuickSilver Technology, Inc. (none*)	Dynamic Logic for Mobile Devices	12/29/00				
SiRF (none*)	Silicon for Wireless RF, GPS	12/29/00				
Taiwan Semiconductor (TSM†)	CMOS Semiconductor Foundry	5/31/01	19.86	9.49	8.39- 19.02	31.9B
Tensilica (none*)	Design Environment Licensing for Configurable Soft Core Processors	5/31/01				
Transmeta (TMTA)	Microprocessor Instruction Sets	12/29/00	23.50	1.41	1.25 - 50.88	189M
Triscend (none*)	Configurable Microcontrollers (Peripherals)	2/28/01				
United Microelectronics (UMC†)	CMOS Semiconductor Foundry	5/31/01	10.16	5.32	4.42 - 12.23	12.2B
Wind River Systems (WIND)	Embedded Operating Systems	7/31/01	14.32	10.50	9.71 - 49.31	816B
Xilinx (XLNX)	General Programmable Logic Devices (PLDs)	2/28/01	38.88	23.53	19.52 - 91.94	7.8B

* Pre-IPO startup companies. ** ARK is currently traded on the London Stock Exchange †Also listed on the Taiwan Stock Exchange

NOTE: This list of Dynamic Silicon companies is not a model portfolio. It is a list of technologies in the Dynamic Silicon paradigm and of companies that lead in their application. Companies appear on this list only for their technology leadership, without consideration of their current share price or the appropriate timing of an investment decision. The presence of a company on the list is not a recommendation to buy shares at the current price. Reference Price is the company's closing share price on the Reference Date, the day the company was added to the table, typically the last trading day of the month prior to publication. The authors and other Gilder Publishing, LLC staff may hold positions in some or all of the companies listed or discussed in the issue.

Ask Nick: Don't forget, all subscribers have exclusive access to Nick on the DS Forum. Just enter the subscriber area of the site and log on with your questions or comments.