WP213 (v1.1) July 21, 2004

Comparing and Contrasting FPGA and Microprocessor System Design and Development

By: Karen Parnell, Roger Bryner

Programmable Logic Devices offer a cost effective alternative to custom microprocessors due to their generic nature with the added benefits of short time-to-market, no NRE costs, off-the-shelf availability, ability to control inventory in peak and trough times, and ability to reduce total cost of ownership over the lifetime of an end product.

Microprocessor obsolescence is a major concern for many companies. Programmable logic can provide a viable solution to this problem. By using soft core microprocessors embedded within a programmable logic device, not only can you own the processor core for use in any future devices and platforms, but the design can be both flexible and scalable to suit different platforms.

This paper compares and contrasts FPGA and microprocessor system design and development flows with the aim of helping the designer and definer of state-of-the-art electronics systems to make a considered and well-informed architecture decision. The first part of the paper introduces Xilinx and PLDs to people new to programmable logic. Next, the paper describes the benefits PLDs can bring to total cost management, technical advantages of PLDs, and microprocessor design flows and methodologies. The paper then discusses future trends, the use of soft core microprocessors, and how these can be used to mitigate against processor obsolescence.

Introduction to Xilinx

Xilinx invented Field Programmable Gate Arrays (FPGAs), holds multiple patents, and is the clear market leader in programmable logic in terms of both revenue and technology. Xilinx was the first programmable logic company to exceed one billion dollars in FY2000 and remains the leading innovator of complete programmable logic solutions. Founded in 1984 and headquartered in San Jose, California, Xilinx fulfills more than half of the demand for these devices today (see Figure 1). Xilinx programmable logic provides a revolutionary alternative to custom logic chips that require weeks or months of design time. As a "fabless" semiconductor company, Xilinx does not own or operate silicon wafer production facilities. Rather, the Company forms strategic alliances with chip manufacturers. This strategy allows Xilinx to focus on research and development, marketing, and technical support, while having access to the most advanced chip processing technologies currently available.

Company History and Financial Summary

Xilinx leads one of the fastest growing segments of the semiconductor industry—programmable logic devices (PLDs) with 51% market share, shown in Figure 1. PLDs represent an exciting growth potential in the semiconductor market thanks to their flexible nature and ability to change functionality, even after being manufactured. Gartner / Dataquest forecasts the PLD market to grow to $5.6 billion by 2006.

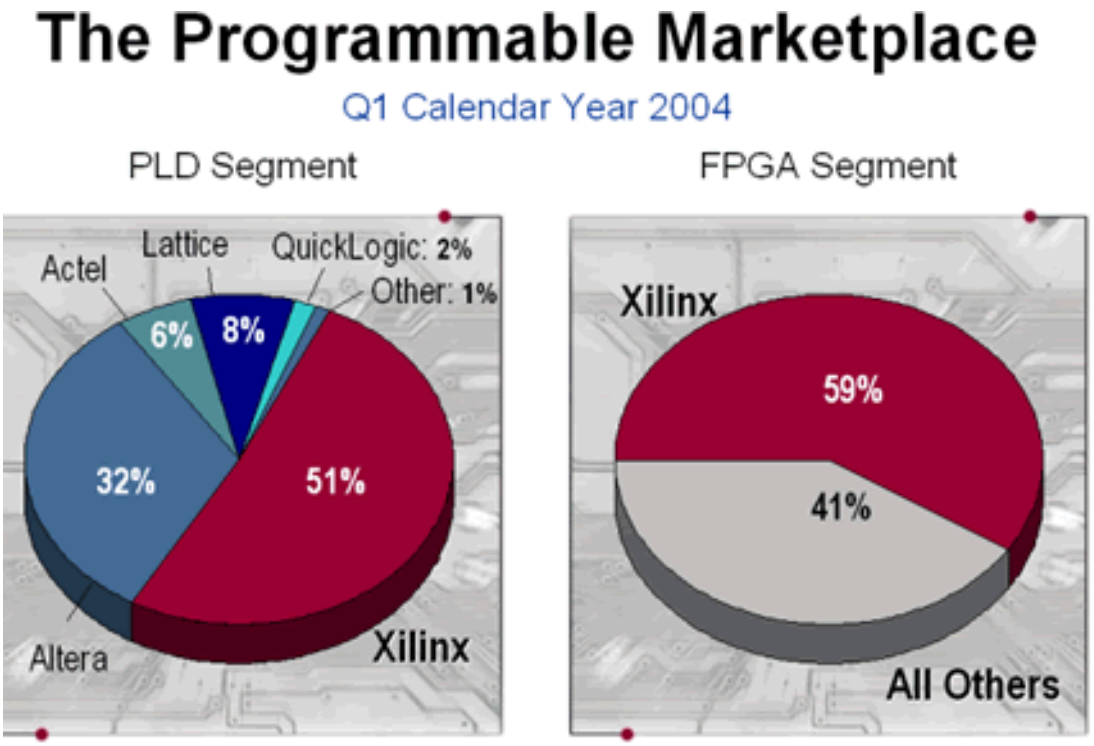


Figure 1: **The Programmable Marketplace**

Market Leader

In the larger $16 billion Application-Specific Integrated Circuit (ASIC) market (which includes PLDs), Xilinx ranks as the world's sixth largest supplier and is the only PLD company in the top ten. Eighteen years ago Xilinx established the fabless

semiconductor model, outsourcing everything but the design, marketing, and support of its products. Xilinx founder and board member Bernie Vonderschmitt is the 2002 recipient of the Fabless Semiconductor Association's (FSA) Morris Chang Exemplary Leadership Award for his pioneering vision of the fabless business model. Almost every new semiconductor company founded in the last 5-10 years has utilized the fabless model.

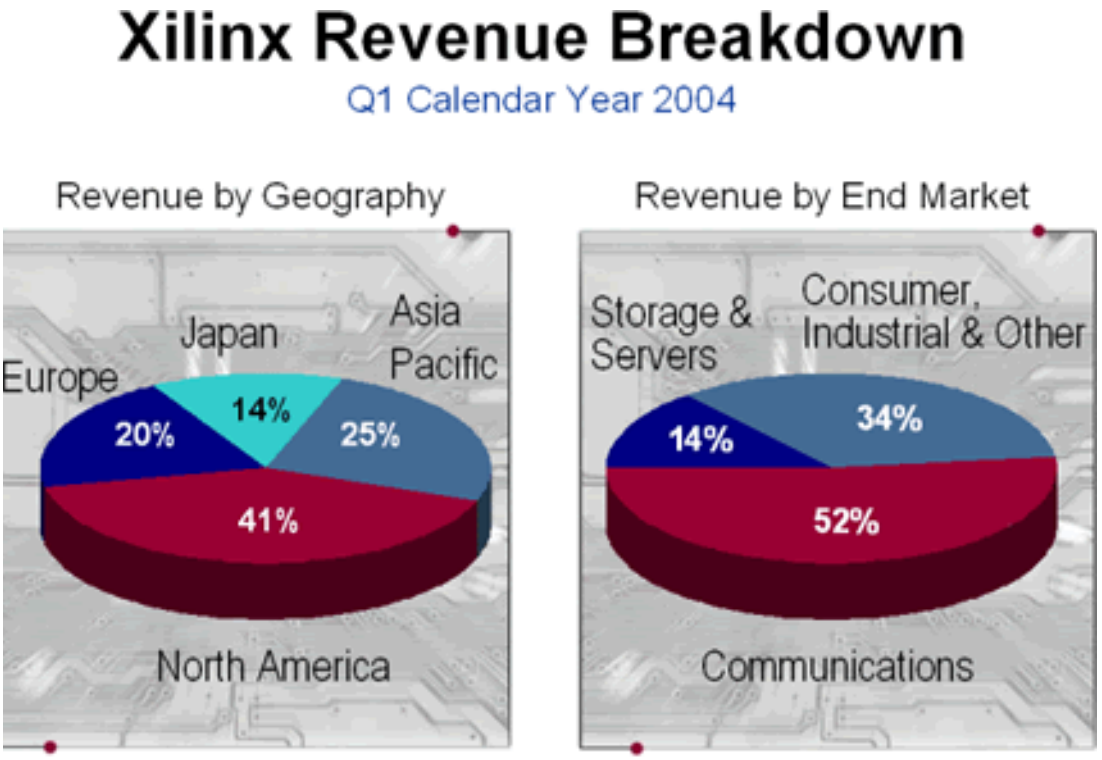


Figure 2: **Xilinx Revenue Breakdown**

For fiscal 2003, Xilinx reported revenue of $1.16 billion, net income of $125 million and has zero debt on its balance sheet. Xilinx became a publicly traded company in 1990 (NASDAQ: XLNX) and has since generated positive free cash flow. Since its initial public offering, Xilinx has granted stock options to all employees, with about 85 percent of the options going to non-executives. Headquartered in San Jose, California, Xilinx has significant operating facilities in Colorado, New Mexico, Ireland and Scotland. Xilinx (XLNX) is traded on the NASDAQ and has a market capitalization of $9.9Bn (28 Oct 2003).

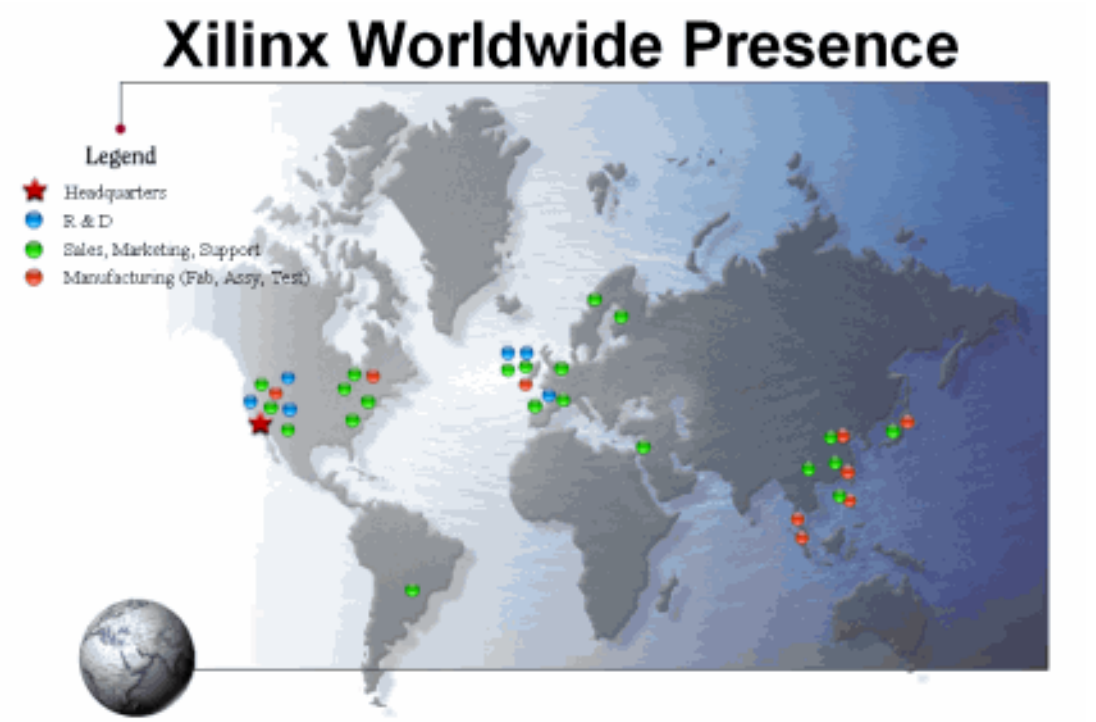


Figure 3: **Xilinx Worldwide Presence**

International sales account for approximately 59% of the company's revenue.

Products and Services

In the digital world, there are three types of electronic chips: microprocessors, memory, and logic. Memory chips are used to store information. Microprocessors and logic devices are used to manipulate, or interface with, the information contained in memory. Programmable Logic Devices (PLDs) are "off-the-shelf" logic chips that the customer, rather than the chip manufacturer, programs to perform a specific function. With the ability to program their own chips, customers realize two key benefits: product design flexibility and faster time-to-market. Given today's shorter product life cycles, both of these factors can be critical determinants of a product's ultimate success. Electronic equipment manufacturers rely upon PLDs to make fast design changes, accommodate uncertain production volumes, and accelerate the introduction of their products to the market place.

Customer requirements for logic solutions that provide higher speeds, greater logic density and integrated system-level functions continue to drive the demand for Xilinx products. The company currently offers several series of FPGAs and Complex Programmable Logic Devices (CPLDs) that are tailored to meet the requirements of different applications and are often cited as industry-leading solutions. *EDN*, *Electronic Products*, and *EE Product News* have all honoured Xilinx with product-of-the-year awards. Most recently, *Electronic Design* named Xilinx's introduction of the SRAM-based FPGA one of the electronic industry's top 50 milestones in the last 50 years. Full details on all Xilinx products are available on the Web at: www.xilinx.com.

Xilinx produces two major lines of programmable products, each based upon a different technology. A non-volatile flash-based process is used to manufacture CPLD

parts. The memory used to store the logical design is stored in on-chip flash memory that can be reprogrammed by the end user. In the volatile CMOS-based FPGA devices, the initial state of the FPGA is unknown and data has to be read from an external non-volatile source such as flash to initialize the FPGA. This paper deals with the second type of device, the FPGA, as the cost of this solution has been falling and the complexity of designs this technology can handle includes soft processors. Specific emphasis is placed on the current Spartan-3™ product line, which is the lowest-priced and highest-volume FPGA technology currently available.

Advantages of Programmable Logic in Future Automotive Designs

This section is split into two main sections, the first being advantages of PLDs with respect to total design life cost savings, and the second focusing on technical advantages.

Vendor Design Costs

During the planning stage of any large project, many considerations and decisions must be addressed that can severely impact cost and project time scales. A recent study by the Ford Motor Company attributed 70% of the project overall cost to decisions made during the design process (shown in Figure 5). Thus, time spent selecting the correct and appropriate system architecture is time well spent. At the heart of all electronic systems is a processing engine. The choice of processing engine is extremely important because this directly impacts time spent in design, verification, test, production, and costs associated with inventory management, field enhancements and maintenance, and end-of-life management. This section will detail these costs and suggest ways to lower these costs.

Design and Development

Design costs are primarily worker-hour and non-recurring engineering (NRE) costs. PLD designs require fewer worker hours, because design and debug changes can be made instantly, hardware and software can be developed concurrently, test vector generation is unnecessary, and product lead times are practically zero compared to ASICs. PLDs have no external NRE costs, which can be a very important consideration as production volumes per design revision are much lower in rapidly evolving markets. In many cases, Xilinx offers free PLD design tools, thus increasing the PLD cost advantage.

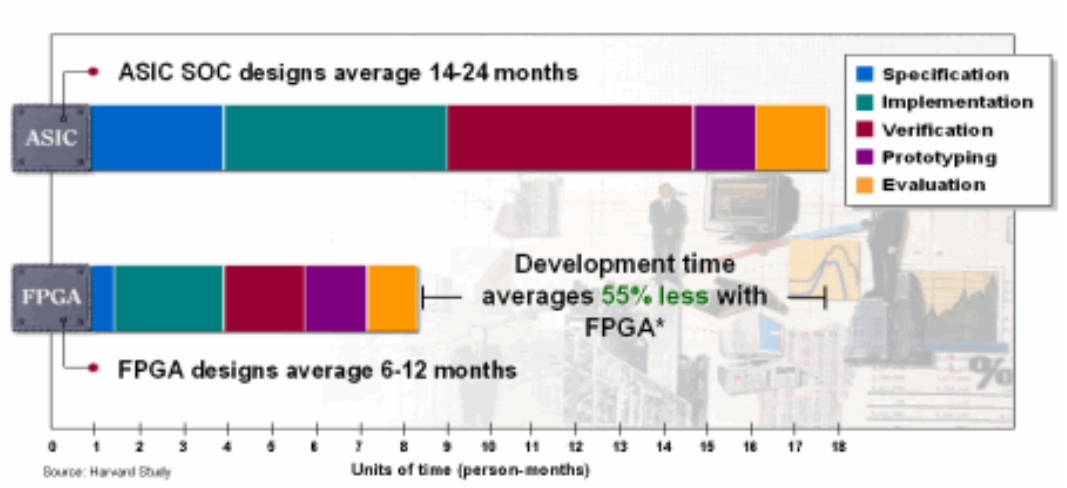


Figure 4: **FPGA vs. ASIC/Custom Microprocessor design time**

There are also lost opportunity costs associated with custom microprocessors. Systems using custom processors lose market penetration due to long development times. Furthermore, the inflexibility of custom processors based systems prevents upgrading products to meet changing market needs. Overall, PLD design cost advantages alone can often outweigh any custom processor unit cost advantage.

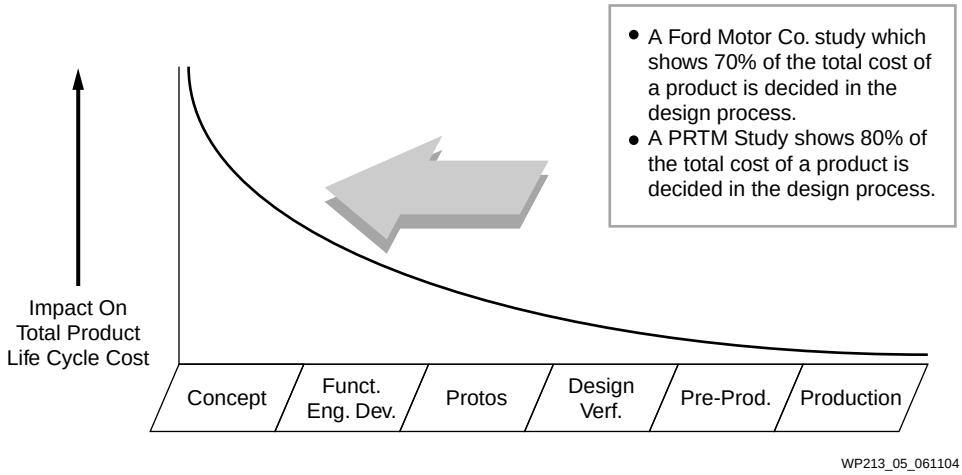


Figure 5: **Impact of Design Process on Total Product Life Cycle Cost *(Source: SMTA Vendor Day, June 4, 2003)***

Device Qualification

Device qualification can be costly and take 9 months or more depending on the end application and temperature range. The cost is usually split between the vendor and the customer but takes time and effort to complete and cannot be undertaken lightly. Once a device is qualified, it is then common practice to open for use in multi-projects to save costs on qualifying other devices. Obviously, the more generic the device, the

more projects it can be used on; and it is very desirable to cut down on the number of different devices qualified. PLDs are ideally suited to this scenario as they are in nature generic, are low cost, and enable design re-use through sharing of Intellectual Property (IP) cores. Customers tend to use a combination of third-party off-the-shelf IP cores, cores designed using the Xilinx Core Generator tool and bespoke in-house designed cores. This in-house IP is what differentiates companies from their competition and once developed can be shared easily between the various design teams. The IP cores are fully verified discrete blocks of logic-based design that can be dropped into many designs and invariably work the first time with repeatable timing and functionality. The Xilinx design tools fully support design reuse of customer IP.

Production

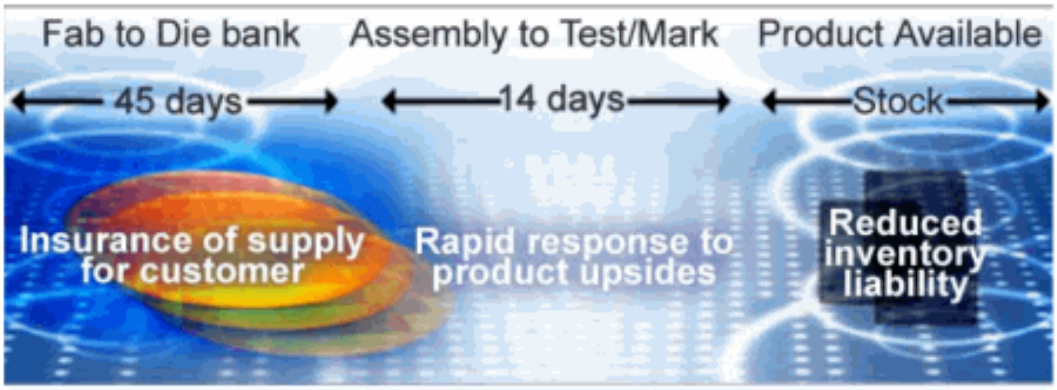
Unit cost is the cost that most people think of first in this category. However, to get a true cost of ownership, customers should consider all costs that are associated with a component. Custom devices usually do hold the unit cost advantage over PLDs, but this cost differential is shrinking while fixed costs for custom processors are rising.

Today's PLDs have aggressively driven down both die size and package costs to provide a persuasive custom processor alternative. The advanced system capabilities of modern PLDs allow designers to integrate discrete component functions into the PLD—reducing board and total component costs.

Finally, unlike an inflexible custom processor, one line of PLDs can be inventoried to supply multiple applications. You can reprogram and redeploy PLDs as needed. Combined with much lower minimum order quantities, PLDs reduce inventory costs, as well as the risk of obsolescence, further offsetting PLDs' unit cost disadvantage.

Inventory Management

Inventory management is a very effective way of reducing overall costs and reducing exposure to dead capital. Traditionally with custom products, the minimum order quantities (MOQs) are high, based on large lot sizes, they have longer lead times (up to 16 weeks – built to order), demand forecasting is difficult, end-of-life inventory balancing is difficult and over-ordering may lead to scrap. With a generic PLD, MOQs are very low (typically less than 100 pieces for FPGA), and products are usually in stock, so it is easier to match run rate with lead time. End-of-life management is much simplified as the MOQs are low, so you need to order only what you will use, and products can always be re-used or used on other projects. PLDs also make it much easier for customers to react to sudden upsides or downsides in demand.



Inventory Management	Custom Product	Xilinx Programmable Logic
End of Life Inventory Balancing	Difficult / Scrap	Simplified / Re-useable
Minimum Buys	Lot Size Minimums	Low / None
Product Liabilities	Very High	Lower
Product Availability	4-16 weeks	Stock to minimal
Forecast Flexibility (Up / Down)	High Risk	Lower Risk
Lead Time vs. Run Rate	Difficult Match	Easy Match

Figure 6: **End-of-Life Costs**

Life Cycle

Life cycle costs are probably the least considered costs when designing an electronic system, but they can have a significant impact on the total return from a given design. Using PLDs instead of custom devices means practically no risk of obsolete inventory, because standard-product PLDs can be redeployed to different applications as needed. Unlike custom processors, PLDs do not have to be scrapped if a specific project is cancelled.

In addition, because PLDs can be reprogrammed, it is possible to upgrade products in the field simply by downloading a new hardware configuration file to the PLD. Bug fixes and feature upgrades can be performed remotely without changing any hardware.

Finally, long-lived legacy products using custom processors can face the need for board redesign should the custom microprocessor or process be discontinued. PLD reprogrammability avoids this problem. Thus, legacy products with PLDs do not siphon design resources from next-generation products, and this presents a significant market cost savings.

We must look beyond unit cost when deciding on logic solutions based on cost. In many applications, the unit cost savings of ASICs and custom processors are more than offset by the lower risks, design, production, and life cycle costs of PLDs.

Mask Costs

There are number of issues involved when planning and designing an integrated circuit. One is the time that it takes to get a product to market. Integrated circuits require many hours of development and testing. To build a new prototype microprocessor and put it into production can cost as much as $16 million. In the past, the costs of this effort were lower and the expected unit costs had not declined to their

present point, and with each passing generation the design and mask costs increase and custom silicon of any form becomes more expensive. If difficulties are encountered once the chip has been produced, the costs of reworking the design for the developer can be debilitating.[1] With mask and design cost escalating, can manufacturers continue to make custom processors for one customer?

Mask costs alone can cost as much as $1 million for 130-nanometer, and for 90-nm processes, $1.5 to $2 million. For many custom processors starts, these escalating costs can be prohibitive. Costs of System On Chip (SOC) re-spins are shown in Figure 8. Designers are being driven towards a right first time design scenario, which can impact design times and lead to products being late to market. A study by McKinsey and Company has shown that products that are 6 months late and on budget earn 33% less profit over a 5-year period (see Figure 7).

According to a recent report by iSuppli, they conclude that mask costs are still eclipsed by engineering costs, particularly verification, and are in the range $5 to $10 million.

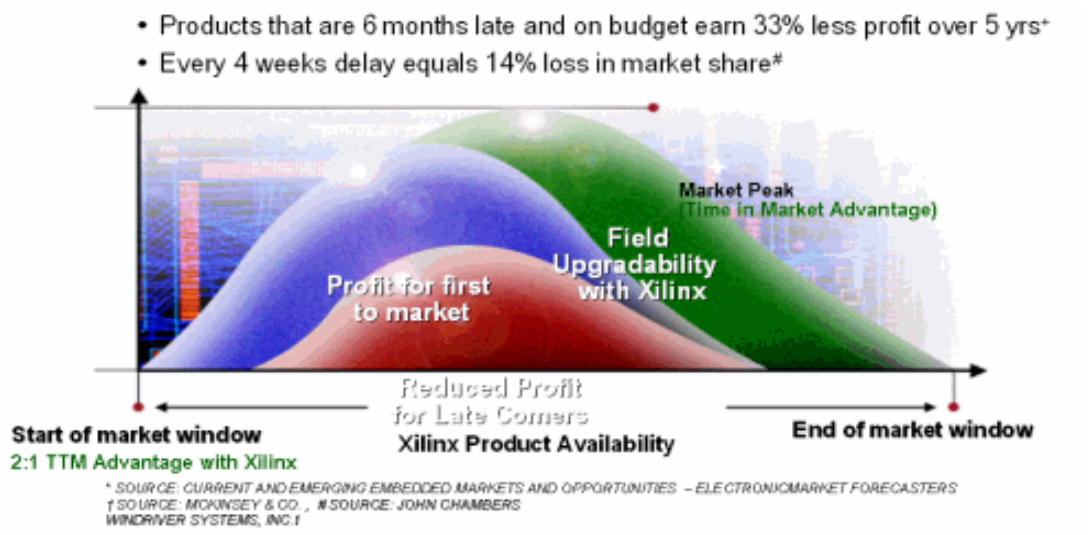


Figure 7: **Time-To-Market Advantage**

The cycle-time impact of a re-spin usually has greater consequences than the cost. Time-to-market in high-technology products is the most important factor in profitability. Gross margins on the first products at a certain performance of capacity level are much higher than the fourth or fifth product to enter any market. Additional funding levels and future initial public offerings often hang in the balance of time-to-market. (Figure 8).[2]

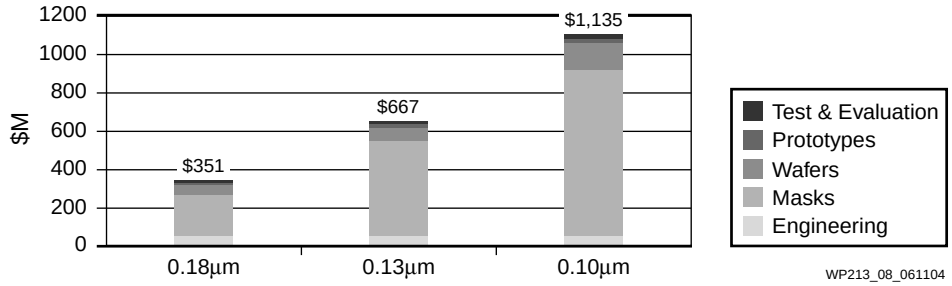


Figure 8: **Cost of a custom silicon device re-spin[2]**

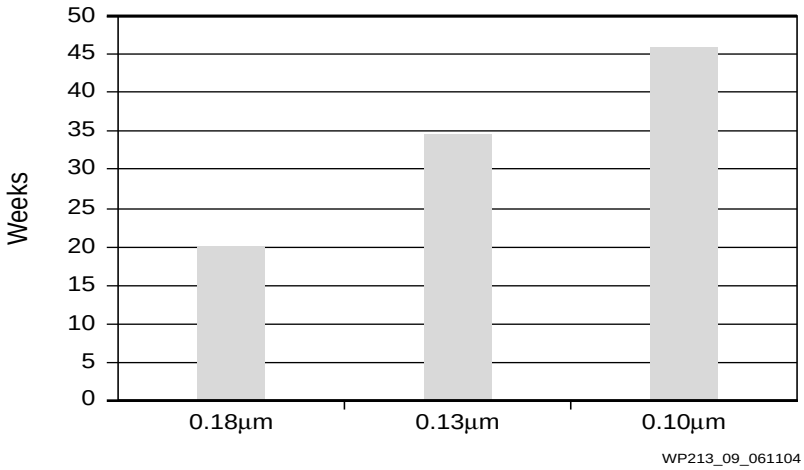


Figure 9: **Schedule Impact of an SOC Re-spin**

Historically Moore's Law enabled semiconductor manufacturers to benefit from economies of scale associated with process geometry migration to get more dies per wafer and hence decrease cost and increase speed. In the past, mask costs were manageable at the geometries larger than 0.18um and re-spins did not impact too heavily on cost and delay to production. A good barometer is the number of ASIC starts; this has reduced from 12,000 in 1995 to 2,000 in 2002[3]. Also, these tend to be on larger geometries to keep the MOQ relatively low with the process node of choice being 0.18um for 2002. This industry barometer is interesting to us as it shows that production of any custom device, be it custom processor or ASIC, is on the decrease due to the escalating cost of NRE and this will only get worse over time.

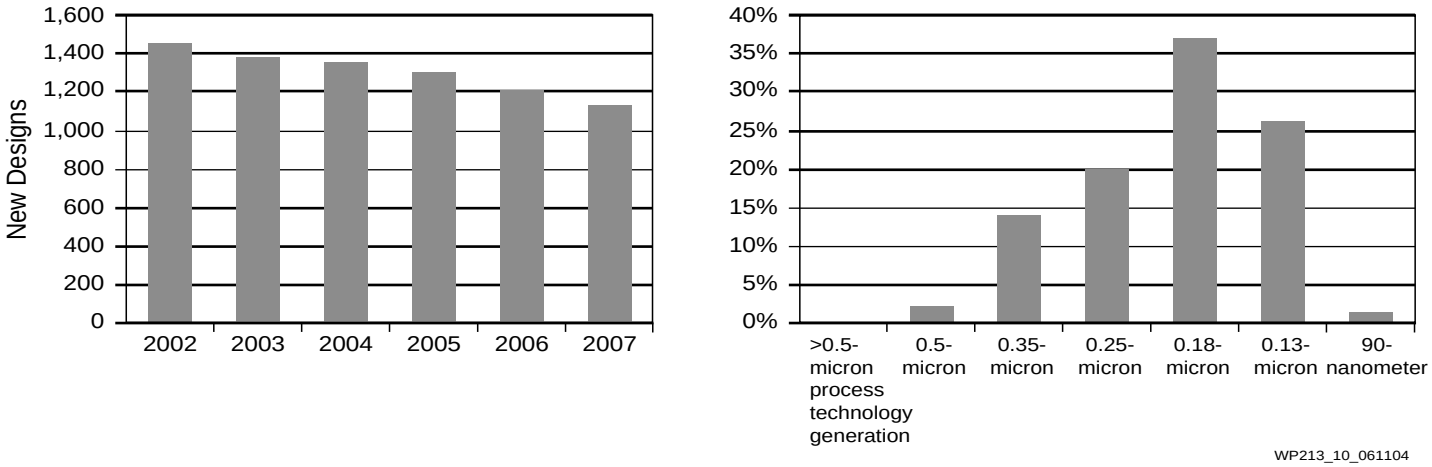


Figure 10: **Custom Silicon Design Starts**

While escalating mask costs and complexity of designs (and their associated lengthening of time to product delivery) are causing custom processor vendors to think twice about each design start, PLD manufacturers can still benefit from lower device costs of smaller geometry designs. Figure 11 below shows that by producing the latest Spartan™ device (Spartan-3) on 90nm process technology on 300mm wafers, we can achieve nearly five times as many die per wafer than one at the 130nm / 200mm combination. 300mm wafers allow for more than twice the usable area, 2.5 times as many die, and a 30% cost reduction.

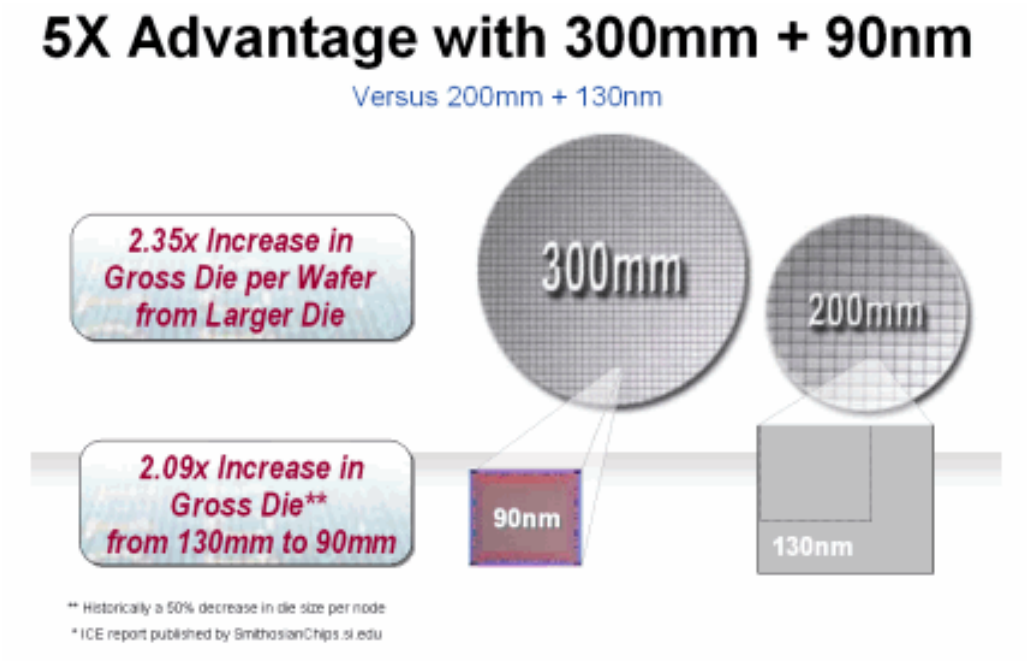


Figure 11: **The Advantage of 300mm wafer and 90nm process**

Technical Advantages of PLDs in Automotive Designs

Programmable Logic Design Concept

Early FPGA chips were used as a replacement for discrete logic chips such as the 7400 series of basic logic functions or simple FIFO or buffer functions. This use has now almost eliminated this type of logic from the board as the cost of packages for functions this small became prohibitive and more complex designs required more and more special purpose logic. Design was originally done in a similar way to how a design using discrete chips was done, with schematic tools that showed simple logic elements attached to each other.

As the capacity of FPGAs grew, the complexity of designs that could be accomplished grew as well. Design tools used in ASIC design found their way into FPGA design. Chief among these tools is the use of hardware design languages (HDLs). Verilog and VHDL are the standard languages used for design of hardware systems at this time at the highest level. Although other approaches such as “polygon pushing” are used in standard INTEL processor and FPGA design to achieve the most optimal performance possible by tailoring every detail of silicon layout to function and performance, the end customer does not commonly use such techniques in either ASIC design or FPGA design. In some cases, IP Core blocks targeted for ASIC or FPGA are optimized beyond the simple VHDL or Verilog description, but in such cases an adaptation layer and treating the IP as a “Black Box” allows the IP to be used in conjunction with user designs.

FPGAs are un-programmed when they first receive power. An external non-volatile memory is required to hold the design bits and stream them to the FPGA to boot it. This process is very fast and happens in a fraction of a second. The amount of data involved can range from a fraction of a megabit to several megabits depending on the size of the FPGA used. The speed of programming FPGA parts can be as high as 264 megabits per second, resulting in boot times that are adequate for the most demanding applications.

One option for programming is to use a dedicated serial FLASH device from Xilinx known as a *configuration PROM* (Programmable Read-Only Memory). A wide range of sizes match FPGA requirements as shown in Table 1 below.

Table 1: **PROM sizes for Spartan-3 Devices**

Spartan-3	Bits	PROM
XC3S50	326,784	XCF01S
XC3S200	1,047,616	XCF01S
XC3S400	1,699,136	XCF02S
XC3S1000	3,223,488	XCF02S
XC3S1500	5,214,784	XCF08S
XC3S2000	7,673,024	XCF08S
XC3S4000	11,316,864	XCF16S
XC3S5000	13,271,936	XCF16S

Even larger memories can be supported when more than simply an FPGA must be programmed. External FLASH, SRAM (Static Random Access Memory), or any memory requirement can be met in addition to FPGA programming requirements after FPGA programming is complete. Memory used to initialize such systems must be memory shared by processor file systems or networked file systems. Host programming allows for Xilinx FPGAs to be programmed from a host processor and any accessible file system, communication port, or any other IO conceivable. This is the most flexible and general system for programming.

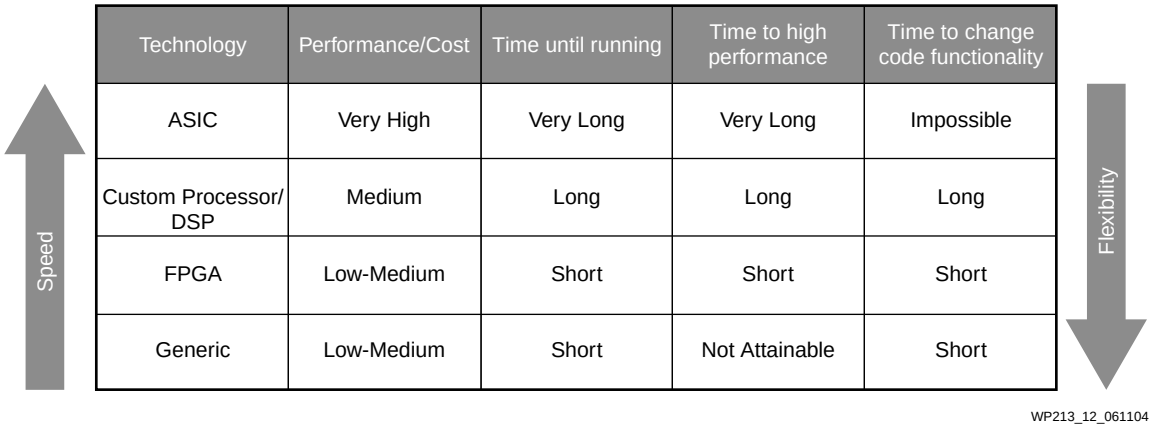


Figure 12: **Design Choices[4]**

Figure 12 summarizes the design choices associated with electronic module design. The choice is dependant on time-to-market requirements, speed of design, cost, performance, and whether the design needs to be changed over time. In this example, *ASIC* is defined as a product designed and used by one customer, *custom processor/DSP* is defined as a product designed for a particular application but with many end customers, and *generic* is defined as a general purpose off-the-shelf microprocessor with multiple uses and customers.

FPGAs as Generic Devices

Programmable Logic Devices by their very nature are generic; that is, they can be considered as “blank digital design platforms” that can be given a personality appropriate to the piece of electronic hardware in which it resides. These end applications can be as diverse as telecommunications base stations through to digital televisions and in-car infotainment systems. This means that the very high development costs associated with semiconductor devices can be spread over many thousands of customers (unlike custom processor mask charges being borne by the customer it has been designed for). Not only can customers benefit from economies of scale, but as the silicon itself is not tied to one end market it is less likely to be pruned or obsoleted. The same generic FPGA is supplied to automotive, consumer, telecommunications, and other end markets. This means that the PLD is not as affected by the cycles of end markets that have shown historically to have peaks and troughs at different times.

Enabling Platform Designs

Recognizing the huge advantage of design re-use and economies of scale, the electronics industry is adopting a “platform” design approach. This means the design

and production of one PCB or platform that can be used in many projects. The design philosophy has lead to scalable and even upgradeable solutions, which can meet evolving and changing standards, or appreciation that in future designs might need to be upgraded to include new features for the end customer to enjoy. By adopting a platform approach, the same PCB can be used for low-end, medium and high-end products. This can be achieved only if the design architecture is designed around a product range that has the same package and I/O configuration across multiple densities. All of the Xilinx Spartan FPGA families have cross-over packages that span at least three densities. For example, a module could be laid out to accept an FG256 package part. For the low-end model, a Spartan-3 XC3S200 (200k gates) could be used; for the mid-range model, a Spartan-3 XC3S400 (400k gates) could be used; and for the high-end, a Spartan-3 XC3S1000 (1 million gates) could be used—all in the same package.

Design Integration

To reduce overall product costs, a route favored by many companies is to devote time to reducing overall component count and reduce the number of devices on the Bill of Materials (BOM). Design integration (that is, the process of reducing component count by putting the effective system onto one chip) is one such method. Currently, this can be achieved by integrating many device-level functions into a single ASIC, trying to put as much functionality into software as possible or integrating many functions into an FPGA. As discussed already in this paper, we can see that while in the past the ASIC route was the natural choice, this route is reserved for those who have huge production runs and are not impacted by time-to-market pressures. Putting functions into software is attractive, but in some cases software testing and debug can take as long, if not longer, than the design itself. Choosing the correct balance of processor peripherals might also be difficult because without going to a bespoke processor it is difficult to avoid wasted functions and hence wasted silicon.

Now that the piece part cost of FPGA silicon is approaching that of the equivalent ASIC, more and more companies are choosing to integrate their designs into FPGA.

Design integration delivers:

- Reduced BOM
- Reduced device qualification expenditure
- Reduced cost of order and inventory management
- Reduced stocking costs
- May reduce PCB complexity and hence PCB cost

FPGA design integration also delivers:

- Faster time-to-market
- Same PCB for many projects
- Flexibility to change designs at any stage—even in the field
- End-of-life cost control

Catering for Design Changes

The development of new products is a proverbial mine field for electronics designers. For example, choosing the correct data bus is crucial to the success of integrating and testing units in production and long after the product has rolled off the assembly line. As an example, in the automotive market many components are supplied by vendors who must worry about interoperability. For Tier 1 suppliers and aftermarket unit design companies, the problem is amplified, as they will potentially supply units to

many OEMs who will invariably all opt for different feature sets, data busses, and protocols. The industry has seen a huge shift of design philosophy away from designing a different unit for every OEM and indeed every car model to reconfigurable platforms. Reconfigurable platforms cleverly partitioned between software and reprogrammable hardware will allow you, the designer, to change your choice of system bus or interface late in the design process and even in production. The reconfigurable system concept also enables different standards and protocols to be tried, tested, and put on road trials; and if you don't find them to be suitable, you can just load in another bus interface and try it out until you find the best configuration. While this might seem like a designer's nirvana and unobtainable, it can be realized today by utilizing programmable logic devices in the form of FPGAs and CPLDs. PLDs hand back to the designer the control over all phases of design from prototype, through pre-production and all phases of production. Once the programmable logic based unit is on the road, it can even be reconfigured remotely via a wireless communication link to allow for system upgrades or extra functions. This flexibility and control can be lost when developing systems based on ASICs and ASSP devices. For the purposes of this paper, ASICs are defined as custom products designed and paid for by one customer, and an Application Specific Standard Part (ASSP) is a custom product for a specific application designed for use by more than one customer. Custom microprocessors made at the request of high-volume customers are essentially equivalent to ASSPs.

It is clear that there are numerous emerging bussing systems providing data and control signals in each market segment served by processors. In automotive, low-cost, relatively low-speed busses (e.g., LIN and CAN) and high-speed, real-time data transfer over optical media (e.g., MOST) compete for node to node connections. In instrumentation, simple current loops, serial protocols, and Ethernet are all used. Telecommunications and Military applications use a very diverse set of protocols based upon numerous historical constraints, and changes continue even after product is shipped. Many large customers are backing more than one standard due to uncertainties over which one will eventually prevail. For a designer, these uncertainties can delay development and ultimately lead to lost revenue. A solution that designers are turning to is reconfigurable systems based on FPGAs that can be reprogrammed to accommodate changing standards and protocols late in the design process and even in production.

FPGA On-Chip Memory Resources

FPGAs are SRAM memory-based in nature, which allows for the ability to embed several types of on-chip memory as well as access to off-chip memory resources. Traditionally, FPGAs have been used to interface directly to external high-speed SRAMs and DRAMs. However, with the introduction of the Virtex™ and new Spartan families of FPGAs, with memory features such as on-chip Distributed SelectRAM and Block SelectRAM, with high speed SelectI/O™ and on-chip clock delay-locked loop (DLL) circuits, FPGAs now provide a comprehensive selection of embedded memory solutions.

Xilinx FPGAs provide dedicated blocks of true dual-port RAM, known as *Block SelectRAM memory*. This provides an effective use of resources without sacrificing the existing distributed SelectRAM memory or logic resources. The Block SelectRAM memory is fully synchronous for easy timing analysis and is easily initiated at configuration.

The blocks are very flexible and can be combined to create wider or deeper memory. The True Dual-Port Block SelectRAM is capable of creating FIFOs with independent

clocks running up to 250 MHz. Block SelectRAM offers advantages in many networking and automotive applications that require memory updates without delaying read access.

Xilinx FPGAs also benefit form distributed SelectRAM or Look-Up tables configured as small bits of RAM. The distributed SelectRAM can be configured as a single-port 32x1 RAM or a dual-port 16x1 RAM (one read and one write port), or a single-port 16x2 RAM. It provides shallow RAM distributed throughout the chip and is well suited for DSP applications.

For example, a Spartan-3, 3S1000 with 1 million system gates has 432K bits of embedded Block RAM and 120K bits of distributed RAM bits. On top of these resources, it has 18x18 multipliers for real-time DSP applications, Digital Clock Managers (DCMs) for on- and off-chip clock signal management and tuneable device termination using the on-chip Digitally Controlled Impedance (DCI) which can help reduce Electro Magnetic Interference (EMI).

Spartan-3 FPGAs have a vast array of interfaces to off-chip memory, which allows for fast and efficient data transfer from off-chip low-cost memory. Table 2 below lists the current memory interface standards supported in Spartan-3.

Table 2: **Spartan-3 FPGA Memory Interface Support**

Memory Device	Electrical Interface
DDR SDRAM	SSTL 2.5V
DDR II SDRAM	SSTL 1.8V
DIMM DDR SDRAM	SSTL 2.5V
DIMM DDR II SDRAM	SSTL 1.8V
Network FCRAM™	SSTL 2.5V
Network FCRAM II	SSTL 1.8V, HSTL 1.8V
RLDRAM™	HSTL 1.8V
RLDRAM II	HSTL 1.8V
DDR SRAM	HSTL 2.5V, 1.8V
DDR II SRAM	HSTL 1.8V
QDR SRAM	HSTL 2.5V
QDR II SRAM	HSTL 1.8V
SyncBurst / ZBT™ SRAM	HSTL 2.5V

Real Time Systems – Driver Assistance System Example

Figure 13 shows a conceptual diagram of a Xilinx Field Programmable Gate Array (FPGA) in an Automotive Cruise Control (ACC) Driver Assistance System.

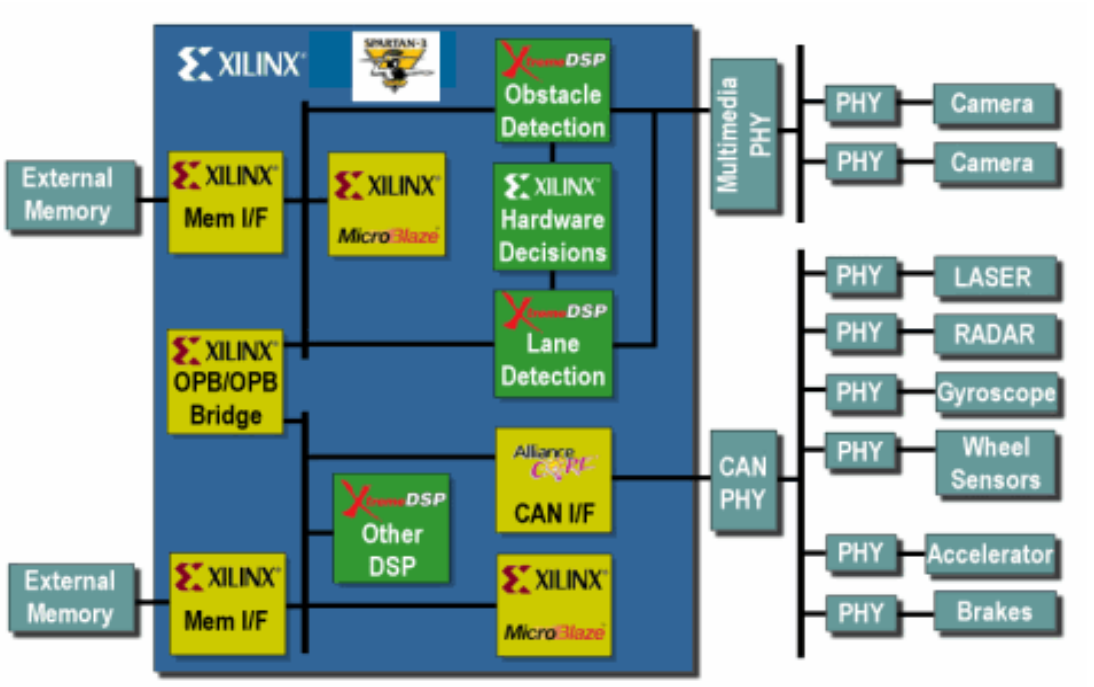


Figure 13: **Concept of FPGA in ACC Driver Assistance System**

The system is partitioned into very a high-speed input processing and relatively low-speed sensor inputs and output control signals, each under the control of its own processor (e.g., a Xilinx MicroBlaze 32-bit soft processor or even an embedded IBM PowerPC in Virtex-II Pro™ FPGAs). The high-speed section is dedicated to the real-time processing of video coming from the cameras mounted at the front of the vehicle. Real-time processing is absolutely critical due to the nature of the application—crash avoidance, emergency procedures, and alerts. Usually, two or more cameras will be used to allow the capture of a stereo image, thus enabling calculation of image depth (directly related to real distances from objects) in the FPGA. When combined with radar and laser measurements, plus the information collected from gyros and wheel sensors to detect motion, a very accurate map of the vehicle's surroundings and its path through them can be calculated.

Using fully flexible FPGAs, as opposed to off-the-shelf video components, equipment manufacturers can easily develop unique, optimized edge detection, image depth, and enhancement algorithms that will differentiate the system performance from that of competitors. Capturing and processing this information in real-time requires the use of math's intensive Digital Signal Processing (DSP) algorithms. However, software processing cannot meet the performance requirements, and, although conventional DSP processors are an alternative, it often needs multiple devices to perform such high-speed tasks. Even ASSP video processors often cannot compare to the extremely high-speed DSP performance of Xilinx FPGAs, also known as *XtremeDSP™ processing*. This is the result of FPGAs having a fundamentally parallel processing structure, rather than being limited to the traditional serial processing found in traditional DSPs and software (comparison in Figure 14).

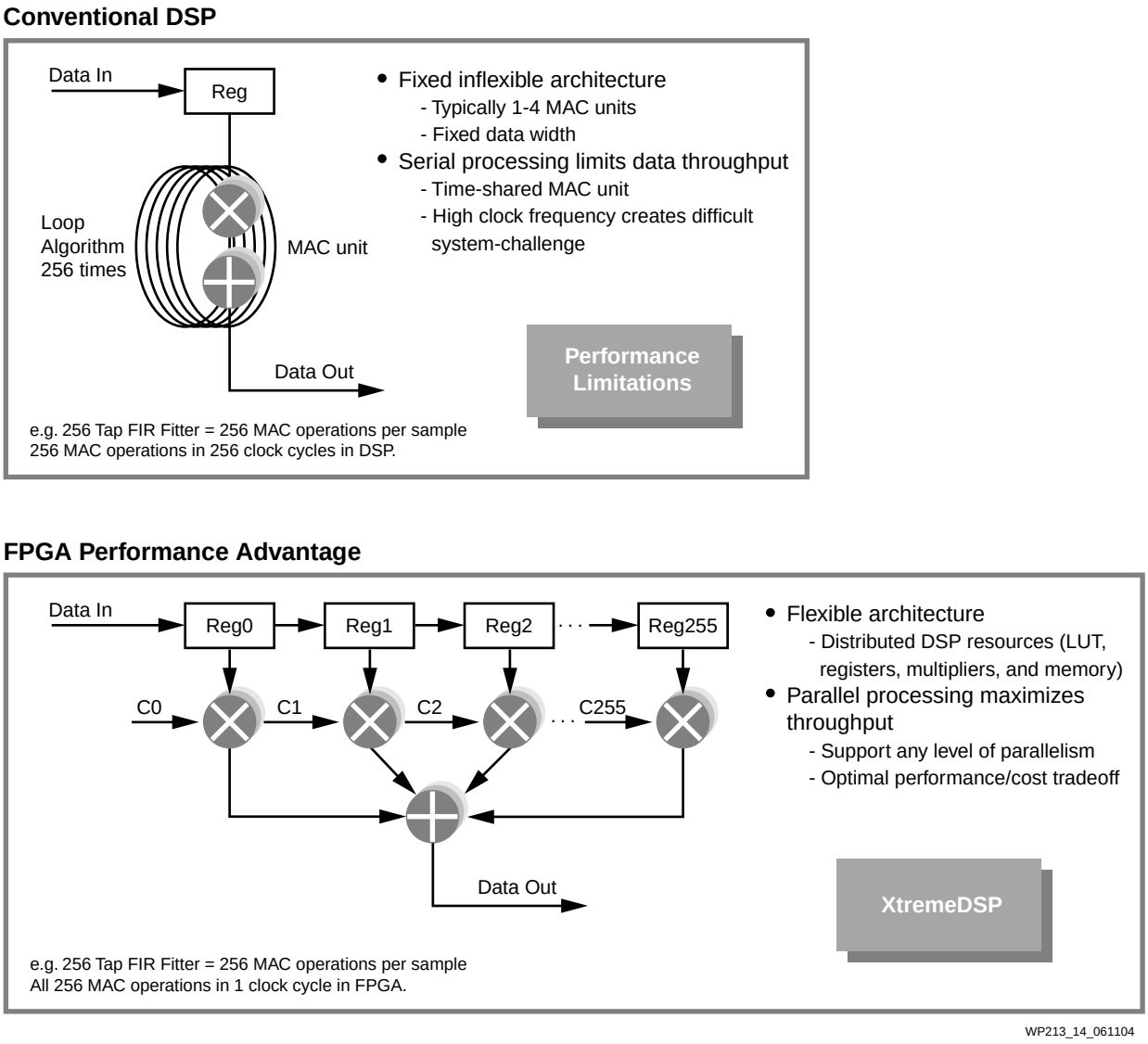


Figure 14: **Xilinx FPGAs Offer Unrivaled DSP Performance**

After processing the video, the decision tree mechanisms can be partitioned between speed-critical algorithms (like sudden object avoidance) and lower-priority algorithms (such as sounding alerts and lane drift warnings). Partitioning speed-critical processes into FPGA hardware also enables several advantages whether dedicated high-performance hardware is used or simply another MicroBlaze processor running a dedicated application. First, the partitioned function will run without interference or interfering with other functions. Often overlooked is the fact that not only might processor or DSP functions have conflicts with other functions, but that communication of two critical protocols over a single bus might conflict. For this reason, a high reliability connection such as a signal to automatically brake might not be wise to place on a CAN bus that holds numerous other signals or a high bandwidth signal such as video. With an FPGA, an additional CAN or other bus can be placed to avoid the possibility of conflict. This is not possible in software functions sharing a single processor. An added benefit of this partitioning is the ease of testing and migrating these functions to other solutions as unit functions.

As well as real-time performance, the reprogrammability of Xilinx FPGAs also offers superb system flexibility, enabling algorithm upgrades to be made even after deployment. This is important, as current driver support systems are still in the early days of research and development. As edge and object detection algorithms are improved over time, hardware upgrades can be done in a matter of minutes and no board redesign is required. This example illustrates that in applications where real-time parallel digital signal processing is required, FPGAs can be used to great advantage. Not only do they meet the processing speed requirements and price points, but they can be reconfigured where designs differ from car model to car model.

Real Time Systems – Network Security Example

Among the most demanding networking tasks are secure transactions. Two common types of encryption are required to facilitate a secure transaction—public key encryption, and private key encryption. Protocols such as SSL require the use of both public and private encryption algorithms. Several different algorithms are used for private key encryption such as RC4, DES, triple DES, and AES. Most public key encryption algorithms rely on modulo exponentiation of fixed point numbers with variable bit lengths. Private key encryption can be accelerated by providing a dedicated core to perform the complete function or parts of it. Public key encryption can be accelerated by over 10 fold by simply moving an inner loop of the complete protocol into a dedicated hardware function.

An example of how much faster a function can operate in hardware is the implementation of a triple DES encryption / decryption algorithm. Table 3 summarizes the results.

Table 3: **Hardware vs. software implementation speeds**

	Encryption		Decryption	
	Software	FPGA Co-processing	Software	FPGA Co-processing
Elapsed time for 1MN of data	5,558.8 ms	424.8ms	5562.9ms	424.7ms
Cryptography	1.51 Mbps	19.7 Mbps	1.51 Mbps	19.8 Mbps

It can be seen from these figures that, although the hardware implementation gives an impressive 13 times speed improvement, it is still running at only 25% of the theoretical maximum rate. The calculations of the performance of the hardware and software encryption timings are shown in Table 4.

Table 4: **Performance timings**

Platform	Clock Speed	Performance
PPMC750	300MHz	1.2 Mbps
Xilinx FPGA	20HMHz	33.8 Mbps

It may be seen that in this application the performance of the hardware was 28 times faster on a 15 times lower clock.[5]

Existing Microcontroller Subsystem Design Process and Methodologies

The design process for existing final microcontroller subsystems begins with the definition of the function of the subsystem's physical interfaces and function. At the initial stage of development, cost and part availability are not as important as prototyping system function and debugging control algorithms, and thus verifying the specification. The next step is conversion of a prototype into a system that meets the cost and manufacturability targets for the final system using standard or custom microcontroller components. This section will not cover the prototyping of systems but only final design and selection of a solution designed to be shipped.

Design Flow

The design of processor modules generally proceeds much like building a house. Architects select general features and components to be used, and then the process of actually building and verifying a system begins.

Processor Selection and System Architecture

The selection of a microcontroller from the plethora of available controllers is fundamentally a balancing act between several equally important considerations. Peripheral set, cost, performance, and availability are all critical considerations in the selection of a processor, while instruction set can be a secondary consideration.

Instruction sets for modern Reduced Instruction Set Computer (RISC) processors are generally within a factor of two performance of each other if not within a few percent on most tasks measured on a per cycle basis. In addition, modern engineering does not use assembly language to any significant extent or on anything but simple projects today. Higher-level languages that are portable across processors, such as C, make the instruction set less and less relevant as a point of engineering choice or attention. The only impediments to switching instruction sets is the fact that tools for a new processor type might cost dollars or engineering and computer time to install and maintain, or might be totally unavailable in the case of a specific brand of Operating System (OS).

Fixed function hardware can sometimes compete with processors when a specific solution provides exactly the solution needed. In some cases, hardware might exist to perform the required function without user provided design files or software. This can be in the form of a chip designed to perform that specific function, or Intellectual Property (IP) provided for use in ASIC or FPGA solutions. In some cases, this is just a processor with software provided by the IP or chip vendor; in other cases, hardware is used, and mixtures of both are becoming more and more common.

Performance is always an issue when selecting a processor. First and foremost, the processor must execute the code it is required to execute within the design limits required on the module. This can be a bulk measurement of millions of instructions per second, or the time taken in the worst case to handle a specific event. In both cases, faster higher performance processors generally work better but cost more; however, there are limits to the ability of a single processor to respond to events that occur at the same time no matter how high their performance is. Deterministic prioritized operating systems such as OSEK (Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug or Open Systems and the Corresponding Interfaces for Automotive Electronics) help to order software development to guarantee deterministic timing; however, there is really no magic here. If too many things are happening at one time, it is difficult or impossible to guarantee that timing will be met in all cases without failure. One way to eliminate this problem is to use more than one processor. This can be very costly when discrete processors are used; however, in

system on a chip solutions in ASIC or FPGA, the cost of additional processors running small tasks from on-chip memory is minimal.

Availability and cost are always issues that must be dealt with. In general, the cheapest solution that meets engineering requirements is the one that is chosen, but if engineering requirements are not met, then low price is no advantage. In many cases, solutions that are economical in one set of requirements become impractical when engineering requirements expand. The ability of a solution to grow with time and further requirements is referred to as “headroom” and is of value to engineers who must often add new features. Examples of providing headroom are the following:

- The availability of higher-performance parts that are compatible with lower-cost, lower-performance parts in the same product line.
- For long-lived products, the availability and price of silicon over time

A detailed discussion of standard parts, custom microprocessors, and FPGA solutions from this perspective can be found in the "Examining Processor Obsolescence" section. Matching the input and output required by the module specification to the peripherals offered in a specific module drives processor selection in many cases. Types of input and output required include some number of general purpose IO (input/outputs), UARTs (Universal Asynchronous Receiver-Transmitter), SPI (Serial Peripheral Interface), I2C (Intelligent Interface Controller), CAN (Controller Area Network), PCI (Peripheral Component Interconnect), memory controllers, event timers, and a wide range of other products. If a processor does not exist with the required peripherals, the customer has a number of options that include going to the manufacturer and requesting a new custom part, using an ASIC, or using peripheral chipsets driven by a generic bus like PCI.

Design Tools

Hardware design consists of generation of schematics or HDL or RTL followed by board layout files to make the proper connections between the devices on the printed circuit boards (PCBs). The first phase of board layout is placement of the parts on the board in such a way that the connections between chips are easily made. External busses and signals often have to cross each other, requiring multiple layers of board metal. Additional layers of board metal add to the cost of the board. For this reason, significant effort is devoted to proper placement, and customers sometimes even resort to different pin outs on packages that have the same function. Once the board is laid out and manufactured, only programmable components such as processor memory and programmable logic can be changed. If a change in the board is required due to need for a faster processor, for instance, the board must be remanufactured and re-qualified, which can be a costly exercise.

Almost all processor systems use the C programming language these days. Even additional tools such as Real Time Operating Systems (RTOS) are simply programs written in the C programming language. The C programming language is converted to machine instructions by a tool called a *compiler*. Compilers range from free, in the case of the GNU compiler, to many thousands of dollars and always come with a wide range of supporting tools such as software debuggers and utilities for programming board memories. Generally, processors of the same family, such as PowerPC parts can use the same compiler tools over a range of processor choices, but a change in architecture from PowerPC to, for instance, an Intel architecture requires another set of compiler tools targeted to that architecture. Primarily for that reason, supporting a large number of processor architectures can be impractical even if there is an advantage in other areas for a particular architecture. In general, though, one can

move from one processor architecture to another if sufficient time is available and the old set of tools does not need to be retained.

Firmware design is the process of adapting the software environment of the processor to the board that it is currently on. This set of software is commonly referred to as a Board Support Package (or BSP). The processor design tools have to be informed of what types of memory are present and at which addresses. Code to control peripherals must be adapted to the peripheral set and configurations required in the system. This programming code is commonly called a *device driver*. Firmware design uses the C compiler to create an environment that allows writers of applications to have a fully functional system to develop on.

Once a board has been “brought up” to the point that basic software runs on it, the application developers write code that actually performs the functions required by the system. In most cases, this can be prototyped on a PC or some other system and be moved rapidly to a new system as generic software written in high-level languages is used.

Microprocessor-Based Design End-to-End Cost Breakdown

The cost of Microprocessor software design is similar for both standard parts and soft processors such as Xilinx MicroBlaze, at the first level of analysis. The design cost consists of the cost of many engineers working on a complex system to guarantee its performance with an acceptable rate of failure given its complexity and specification. This cost must be repeated periodically due to a number of causes. One cause is the re-engineering of the system for a new product model. Another is the obsolescence of a processor resulting in a requirement to replace ageing and obsolete electronics. The issues involved in processor obsolescence are covered in the "Examining Processor Obsolescence" section and will not be covered here. The other issues involved in software design will be.

Although as recently as the late 1980’s and early 1990’s many simple and small processors did not use the C programming language, with time this changed as software grew over the years. For all current significant designs, C is used, even for most 8-bit processors and all 16-bit and 32-bit. The usage of C leads to portability among processor architectures, and for that reason the task of moving from processor to processor no longer requires the software re-writes that were required with assembly language. The task of verification is now simpler, however, and issues of timing and performance have to be taken into account each time a new processor is chosen, as well as buying, learning, and maintaining separate sets of processor tools. Choosing a processor that avoids obsolescence and allows a migration path to newer compatible processors mitigates this verification effort.

While those studying the difficulty of software development cannot agree on the applicability of object-oriented programming vs. procedure-based programming, two things are clear. First, as the size of a module under development grows, so does the effort involved in developing, verifying, and maintaining it. Second, as the number of modules involved in a complete system grows, so does the effort in developing each individual module, and verifying and maintaining the complete system.

As a result of these two factors, the trend in processor subsystem design is the proliferation of sub-modules. The use of sub-modules eases re-design effort at a minimum, by allowing the reuse of modules from one design to another without changes. There are two limits to this: first, as each additional processor is added, this adds cost to the design of the complete system; secondly, more than one module running on a single processor can result in interference between those modules, and as

the number increases, the probability of interference as well as the number of possible configurations to test increases. The unit cost to put an additional sub-module or function into hardware is smaller than that required to merely carve out a section of software code; however, the engineering effort can be greater. Conversely, in the case of simply putting down another processor in an ASIC or FPGA, the cost of an additional processor is minimal as long as the software effort is clear and the amount of memory required can be accommodated on-chip. One low-cost option is to add an additional processor on-chip in soft processor FPGA designs and run the software required for the sub-module from on-chip memory.

Future Concepts

Soft Processors for FPGA and ASIC

An emergent trend is to move from bespoke microprocessors to soft-core processors embedded within either FPGAs or ASICs. This trend has been driven by the long-term supply uncertainties of companies that provide bespoke microprocessors. This uncertainty is due to their inability to take advantage of new process technologies and geometries, with many providing solutions based on the older 0.5um and 0.35um process node. There is good reason to use this process as it provides customers lower NRE charges and lower MOQs, but how sustainable is this older process? Most semiconductor fabrication companies are moving from older processes to the newer 90nm node and beyond to increase economies of scale and throughput. The following section discusses the soft processor solutions from Xilinx and the benefits, both with respect to overall cost and technical advantages.

Xilinx MicroBlaze™ and PicoBlaze™ Soft Core Processors

Xilinx offers both a 32-bit soft processor core called *MicroBlaze* and an 8-bit solution called *PicoBlaze*. The PicoBlaze processor runs at speeds of 116 MHz, yet occupies a tiny footprint of just 35 Configurable Logic Blocks (CLBs), which is equivalent to $0.13 of silicon area in a Spartan-3 device. The PicoBlaze processor can be targeted for use in both FPGAs and CPLDs.

The MicroBlaze 32-bit soft processor core is the industry's fastest soft processing solution, runs at 150 MHz in the Virtex-II Pro architecture, and delivers 120 D-MIPS. The Spartan-3 maximum frequency for MicroBlaze 32 is 85 MHz, giving a performance of 69 D-MIPS. It features RISC architecture with Harvard-style separate 32-bit instruction and data busses running at full speed to execute programs and access data from both on-chip and external memory. A standard set of peripherals is also CoreConnect™ enabled to offer MicroBlaze designers compatibility and reuse. The IBM CoreConnect bus architecture is an on-chip bus that enables communication between the processor core and its peripherals and is used to connect peripherals for the IBM PowerPC in Virtex-II Pro and also the MicroBlaze soft core processor. Table 7 shows the details of the two soft core processors.

The MicroBlaze Embedded Development Kit, including the soft processor core and a standard set of peripherals, is available from Xilinx and its distribution partners. The kit includes a complete set of GNU-based software tools including the compiler, assembler, debugger, and linker. MicroBlaze Kits that are bought from Xilinx Distribution Partners will also include development boards that support the Virtex™-E, Virtex-II™, Spartan-3™, Spartan-II™ and Spartan-IIE™ series of FPGAs. The table below summarizes the two processor cores. Selected Xilinx FPGAs in the new IQ

Solutions range have been qualified to operate over the -40ºC to +125ºC temperature range and are targeted for use in automotive applications such as infotainment, electronic control units (ECUs), instrument clusters, and driver assistance systems.

The MicroBlaze soft core processor can also be provided as source code for migration to ASIC if it gives economic gains. Therefore, designs can be started in FPGA and migrated to ASIC at a later stage if the design does not require further upgrades or changes in the field. Table 8 shows the speed and effective cost of the MicroBlaze processor today.

Table 5: **Xilinx Soft Processors**

Soft Processors	Architecture	Bus	MIPS/Speed	Size	FPGA Support	Design Support
MicroBlaze-32	32-bit RISC	Harvard style busses 32-bit instruction and data buses	120 D-MIPS 150MHz	950 CLBs	Virtex, Virtex-E Virtex-II, Virtex-II Pro, Spartan-II, Spartan-IIE, Spartan-3	Embedded Development Kit (EDK) – soft processor core, peripherals, GNU-based software tools (Compiler, assembler, debugger and linker)
PicoBlaze 8-bit	8-bit	8-bit address and data busses	83MHz	85 CLBs	Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Spartan-II, Spartan-IIE, Spartan-3, CoolRunner-II	Free of charge reference design – includes application note and assembler

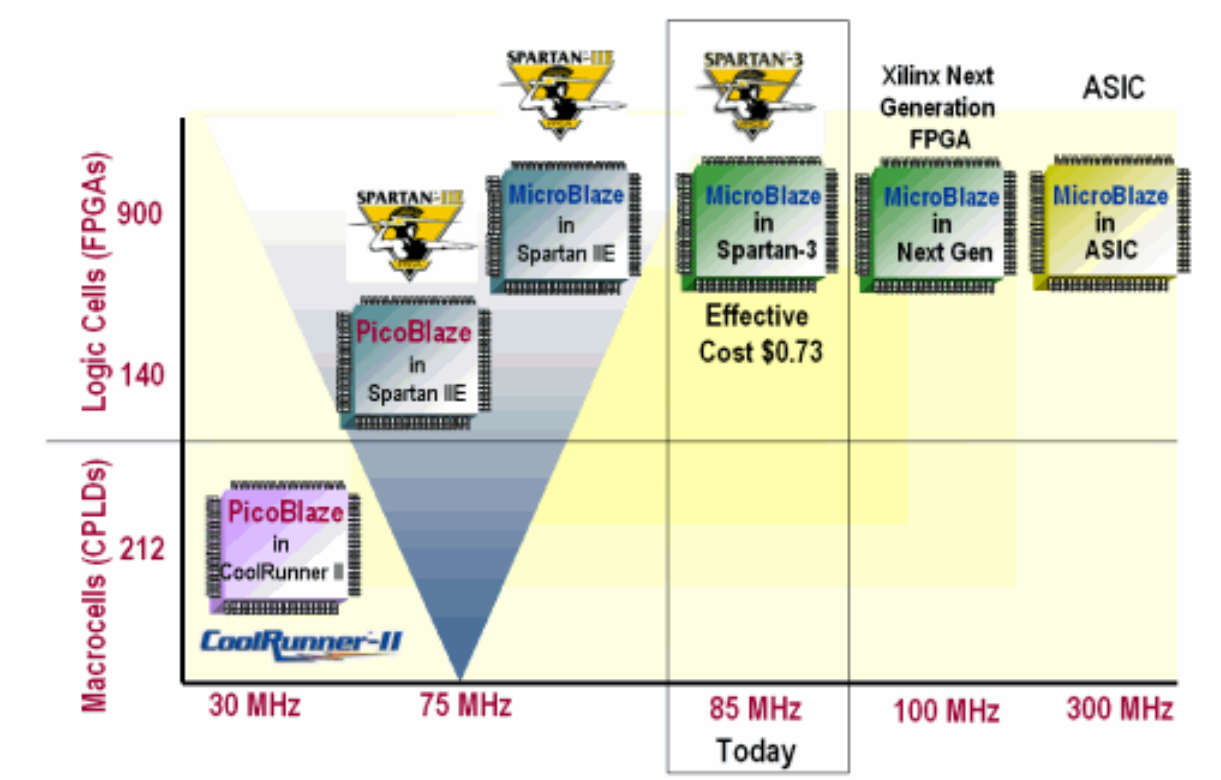


Figure 15: **Xilinx PicoBlaze and MicroBlaze Soft Processors showing size in logic cells and speed in MHz**

Eliminating Processor Obsolescence

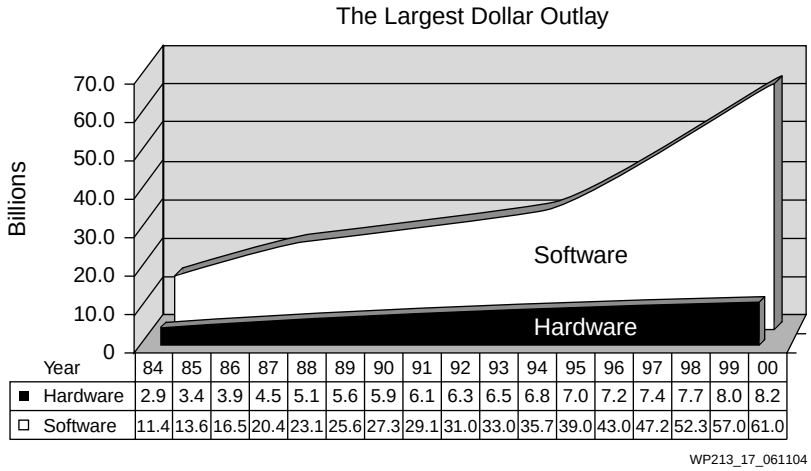
Obsolescence is a concern of most design engineers, and none more so than with automotive electronic module designers. Even though automotive electronics equipment design and development time scales have shrunk recently from 5 to 2 years, the products themselves will still need to be produced for many years and be active in the field for even longer.

The biggest obsolescence headache is that of out-of-date microprocessors and microcontrollers. Processors have shorter life spans than ever and are discontinued at short notice driven by the consumer market trends and the ever present need for speed enhancements. Consumer products such as games consoles and mobile phones have built-in obsolescence to stimulate sales of the latest and greatest products. This built-in obsolescence causes microprocessor manufacturers to chase high-volume sales in new platform introductions, contributing to the problem of obsolescence.

Even if the design has been coded in “C” (which is always promoted as being “portable code”), verification (and, sometimes, architecture-specific) instructions and features can hamper the move from an obsolete processor to a next-generation device. The change-over process is further exacerbated by different package options and I/O configurations necessitating the need for a complete board re-spin. If we imagine the scenario where every Electronic Control Unit (ECU) in a car contains at least one processor and that every car contains up to 60 ECUs, this leads to a major headache every time a processor is obsoleted at relatively short notice.

Cost of Processor Obsolescence

There are several solutions to the problem of processor obsolescence. The applicability of any given solution depends upon a number of variables, including the value of the application software, the projected life of the system, and the amount of time and money available to solve the problem. The most radical and most expensive solution is to redesign the system around a new processor. Depending upon the volume of the code, a redesign can cost hundreds of man-years of time, much of it devoted to validation and testing. Not only is the huge investment in debugging and refining the existing software lost, a clear case of throwing the baby out with the bath water, but the solution is temporary at best. Figure 17 shows the escalating costs of software design over time. If the system has a long projected life, the same problem will recur every few years, as each new design in turn becomes obsolete. Another solution is the last time buy (LTB), which, on the surface, appears to be the most cost-effective option. The problem is that the automotive designer must guess at how much product to buy for the life of their program. A designer who guesses incorrectly is faced with an even more difficult problem—a larger legacy investment that must somehow be upgraded.



Year	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	00
■ Hardware	2.9	3.4	3.9	4.5	5.1	5.6	5.9	6.1	6.3	6.5	6.8	7.0	7.2	7.4	7.7	8.0	8.2
□ Software	11.4	13.6	16.5	20.4	23.1	25.6	27.3	29.1	31.0	33.0	35.7	39.0	43.0	47.2	52.3	57.0	61.0

Figure 16: **Cost of Software Development**

Inserting a new processor along with software written to emulate the obsolete one is presently better in theory than in reality. The concept is appealing and in fact does have some operational history. The legacy software is preserved, and the process is therefore relatively cheap and fast. Once again, the solution is not permanent and, if the system has a long projected life, might have to be repeated every few years. More important, software emulation is inherently a serial process and therefore relatively slow. That means that the new processor must consume much of its performance running the emulation rather than the application. It has been shown empirically that emulation requires, on the average, about twenty clock cycles of the new processor for every legacy instruction it executes. In addition, emulation breeds further obsolescence since the processor used as the emulation engine itself will become obsolete and might force an entire rewrite of the emulator. Figure 18 shows the amount of time as a percentage spent in operations and support, which includes time spent handling obsolescence issues.

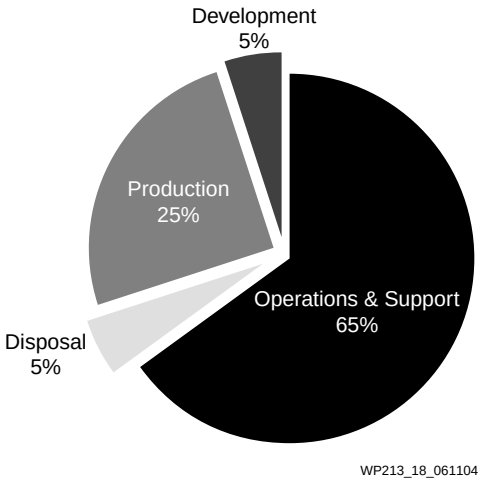


Figure 17: **Life Cycle Costs**

Soft Processor Solution

A radical but robust new solution is emerging to eradicate processor obsolescence and preserve many years of legacy code and development. The new way is to **own** the soft processor core and embed it in FPGA fabric. Not only can you port the core to multiple FPGA platforms (including those that don't exist yet), but you can "design" the peripheral set to meet the exact design requirements, thus eradicating architecture compromises and wasted peripherals.

For example, the designer might desire a processor with perhaps 10 UARTs, an interrupt controller, and access to a block of external FLASH. While many off-the-shelf processors exist that would offer multiple UARTs and the other desired peripherals, they would typically be of sufficient complexity to have numerous other peripherals that would be unused in this system. Not only is the designer paying for the additional peripherals, it is often necessary that unused peripherals in this type of processor have to be placed into a safe mode or otherwise disabled by means of software. This places an additional burden on the software design team, who not only have to make the used processor peripherals operate correctly, but now have to write code for the parts of the processor which are not used. It is clear that purchasing an off-the-shelf solution for this scenario would be highly wasteful not only in terms of initial cost, but also in wasted engineering time during the design process.

With the Xilinx MicroBlaze soft processor, the designer has the luxury of a different approach. They can now start with a processor core and build the peripheral set to meet their exact requirements. Silicon waste is reduced to zero since the designer will only implement what they need. Software design complexity is reduced because no code need ever be written to disable unwanted processor functionality. The creation of unusual processor configurations, which can be changed at any time to suit changes in the specification, is reduced to a simple task.

Even if after ten to fifteen years of field use, when the FPGA hardware might itself be nearing the end of its life, then the soft processor core can simply be dropped into its new FPGA "host" utilizing the same C code and almost all of the same hardware design files as well. The hardware platform might need some PCB modifications, but the legacy code remains usable and intact.

Spartan Product History

All semiconductor products will eventually go obsolete. Xilinx devices are, however, shielded from some of the main reasons for obsolescence. For example, PLDs are generic devices and used in many end markets and by many end customers. Some microprocessors are designed solely for one market and one end application, so if this processor type is not accepted by enough of its target markets, it could be pruned. PLDs do not suffer from this phenomenon as each device has many customers. We can see from the history of the Spartan family of devices that Xilinx devices have a tendency to be produced for many years. Figure 19 shows the Spartan family history and decrease in price points over time.

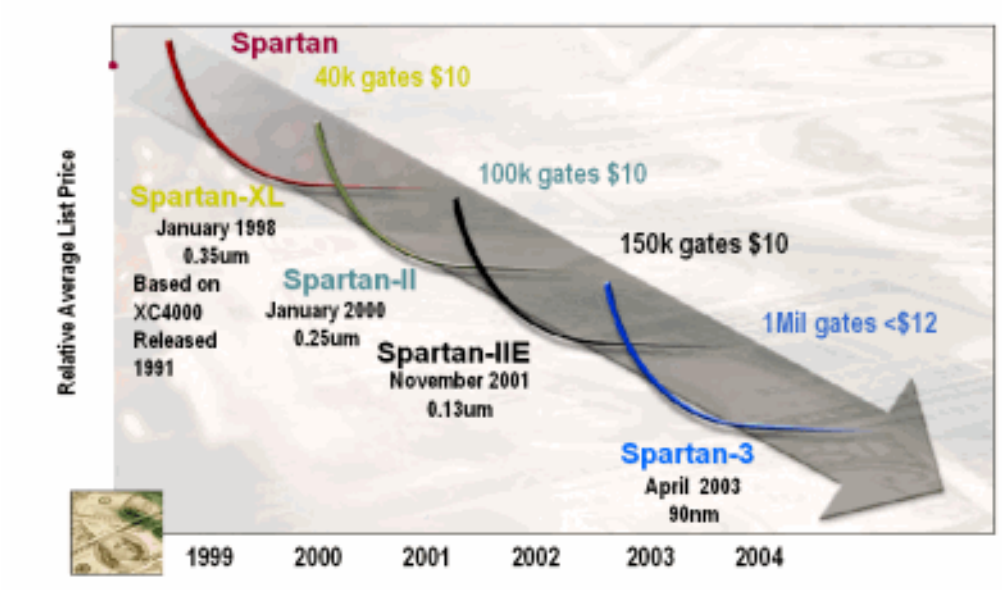


Figure 18: **Spartan Family History**

Figure 19 shows that the Spartan product family was originally based on the XC4000 series, which was introduced in 1991. This architecture was rationalized for low cost in 1998 and released under the Spartan brand. Both Spartan and Spartan-XL™ are still in production today. The fifth generation Spartan-3 was launched in April 2003 being the first ever device produced in 90nm process technology on 300mm wafers.

In the event of device pruning, Xilinx has a structured customer notification process and warns well in advance of this eventuality. This process includes a 12-months' notice of product pruning, a PDN (Product Discontinuance Notification) sent to customers notifying them that there are 12 months to place LTB (Last Time Buy), and an additional 6 months are given in which to accept Last Time Shipments (LTS). History has shown us that this event is rare because there is still high demand for older Xilinx FPGAs.

Technical Advantages of Soft Processor Cores in Programmable Logic

Custom Silicon Design Differences

The design of custom processors using a fixed processor architecture core is no different than the design of a custom ASIC using a processor core such as ARM, ARC,

or Tensilica. The cost of development, as well as any schedule and business impact associated with the development must be passed along to the end customer(s) in order for the customer processor vendor to be viable. While the costs paid by standard parts silicon vendors for masks, layout, engineering, and other factors are unknown and not publicly reported, the costs of ASIC design are well known and have been tracked for several years. It is from these known data points that we can extrapolate trends that affect all silicon chips, whether they are ASICs, ASSP, or standard parts. Generally the only real differences between an ASSP, ASIC, and Standard Part are oriented around business issues; for instance, an ASSP and Standard Part can be sold both to a customer and to the customer's competitors, leading to a lack of competitive advantage compared to ASICs, which cannot be sold to a customer's competitors. On the other hand, the full cost (and risk) of production of an ASIC is born by the customer and the customer alone.

Often, an ASSP is simply a variation on a Standard Part designed to meet the needs of a specific important customer who expresses an interest in a particular set of peripherals and processor core being on one chip. In many cases, the prohibitive cost of designing a custom chip can lead to packaging and software changes to remove function from an existing higher-end part in order to meet the lower price a customer expects. Savings in package costs, typically $0.01 per pin, coupled with savings in design and mask costs can lead to situations where several dollars of discount for a specific part can be economical for the vendor compared to actually producing a new chip. Indeed, the practice of using the same silicon in different packages and selling the highest performance options enabled in software at a premium is a well-established standard parts practice.

The design of FPGA hardware is simpler than the design of ASIC hardware in a couple of ways. First, timing of the FPGA is more deterministic than an ASIC circuit would be, due to the extra resources on the FPGA devoted to routing and timing. In addition, issues of cross talk have been designed out by Xilinx in FPGAs. Finally, the cost and time delay of generating prototype ASICs require lengthy and complicated verification of the design to make sure that the ASIC is free of bugs. In the case of an FPGA, if a bug is found, it can usually be fixed and the FPGA reprogrammed, in most cases even after the product has been shipped!

While generic designs implemented in FPGAs generally do not run as fast as they would in an ASIC using the same process as the FPGA (anywhere from 5-60% slower depending on the specific design), the economy of scale present in FPGAs allows the use of the most advanced chip technology present at the time they are designed, while ASIC flows lag behind by approximately one year or more, increasing performance compared to commonly used ASIC technology.

Hard IP Blocks

For Hard IP blocks or functions present in the FPGA, this advantage is magnified by the optimized design techniques used to gain performance compared to typical un-optimized designs. For this reason, some functions actually run faster and better than those available in even cutting-edge ASIC. Examples include the MGT (Multi-Gigabit Transceiver) function and IBM PowerPC processor of Virtex-II Pro, both of which achieve cutting-edge performance compared to any implementation when measured on speed, power consumption, or size.

The Memory blocks present in FPGAs are optimized in much the same way. They have an additional advantage in that they are programmed as the rest of the part is programmed, resulting in the ability to hold data at boot time such as software code and data. In an ASIC, memories are either non-volatile and cannot be written to, or on

first boot are in an undetermined state and must be written to in order to initialize them.

Many microcontrollers are developed with on-chip flash memory using FLASH (not CMOS) process technology. The introduction of FLASH memory with logic is expensive compared to either alone, and thus the upper limit of the size of FLASH memory parts is a few MBits at most before the cost begins to ramp above $20. When compared with parts containing both FLASH and a microcontroller in memory sizes from 16KBytes to 1Mbyte, the cost of FPGA BRAM memory initialized from off-chip FLASH is comparable to the cost of integrated FLASH memory, but the performance that can be obtained is much greater and the amount of read / write memory cycles is many times greater. Above 1MByte on-chip memory cannot compete with off-chip memory.

In ASSP's, Custom Parts (e.g., custom processors), and ASICs, another consideration is the minimum order quantity (MOQ). As the size of silicon wafers increases, many hundreds to thousands of parts per single wafer are created. A run of a small number of wafers can make no sense when set up time is factored into the cost. For this reason, the MOQ of custom silicon has been steadily growing as has the economical order quantity (EOQ). Even for standard parts, this results in earlier obsolescence for specific configurations of microprocessors compared to FPGAs, which have a small number of configurations, produced in large quantity relying instead on field programming for customization.

Input/Output and Peripheral Set Considerations

Standard parts, whether ASSP's, custom parts, or off-the-shelf microcontrollers, also suffer from a fixed IO pattern. Both the function of the IO pins and their position on the part are fixed. In some cases, this requires complex board routing or placement to accommodate the physical position of connectors or other components that the signals are connected to at the board level. In the case of FPGAs, the pin out of the device can be changed to accommodate the easiest board routing, sometimes resulting in fewer layers of board being required and therefore producing a measurable cost saving.

Perhaps the biggest difference between soft FPGA processors and ASSP's and Standard Parts is the configurability of the peripheral set. The only fixed constraint on an FPGA system is the number of IO present in the chip package and the internal resources. Since each IO can be dedicated to any number of functions, the ultimate in flexibility is achieved second only to ASIC chips, which can also have any peripheral set that can be imagined. One other advantage of FPGA chips is the fact that the available packages range from low-cost low pin count options to very high pin count options delivered at a very competitive price due to volume. In general, the same system including hardware and software is upward compatible with FPGA parts that have more IO pins or more logic or both. Moving a base platform to a larger part becomes a simple exercise rather than relaying out an ASIC. The unique design of FPGAs also allows for the sharing of resources between multiple identical functions. When peripherals are created this way, their size in the FPGA does not increase linearly, but can often be only double the individual case for as many as 16 UARTs, I2C, SPI, or other similar functions.

Multiple Processors in One Device

Perhaps the ultimate freedom of configuration present with the FPGA is the ability to embed multiple processors in a single FPGA. This ability leads to a simulations reduction in part count and therefore pin and package cost, as well as making systems easier to test, maintain, and develop. Many times in software design the testing of single functions is easy, but when many simple functions interact, intermittent

problems emerge that can be hard to debug. By using a separate processor for different tested systems, it is possible to isolate the interface of these systems and ease development and verification. In current FPGA products, it is conceivable to include a MicroBlaze processor, include several dedicated PicoBlaze processors, and eliminate a PCI interface to an off-chip Ethernet device instead placing an Ethernet MAC function inside the FPGA for less than competing solutions. As time goes by, even more powerful and integrated systems can be created for even less cost, mixing and matching systems that used to be on two chips in one.

One Platform Across Many Projects

The ability to create a general-purpose platform that can be used over a number of engineering projects and reconfigured to match the specific requirements of each project is huge benefit. By standardizing on such a platform, engineering time for board layout and time and money spent on making prototype boards are virtually eliminated. Also, familiarization time for engineers getting up to speed on new platform hardware and software is eliminated. Finally, a number of unique tools are available to FPGA designers to help with the development and debug of engineering systems. A tool, known as *Embedded Logic Analyser* (ELA), is available and can be placed in a design running on the chip to monitor any chosen logic signal. The availability of FPGA parts with larger logic and memory capacity in the same package as less costly parts allows engineers to place additional function for debug or development such as the ELA in pin compatible parts. The ability to cross trigger traditional JTAG processor debug from the ELA currently exists in the Xilinx tools, as well as the ability to debug multiple processors and control the ELA from a single JTAG connection.

Conclusion

Choosing the correct devices for the heart of any system is very difficult and is made even harder by the emergence of new technologies such as Field Programmable Gate Arrays (FPGA). The only certainties are that the system will invariably have a processing engine of some type, some memory, and logic. We also know with some certainty that choices made in the design phase impact heavily on the total life cycle costs of the end product, so this decision is key to any major project. This decision will be based on time-to-market, NRE costs, cost of change, obsolescence concerns (and the costs associated with this), and cost to service and maintain in the field.

FPGAs are a viable alternative to custom microprocessors as they can offer a shorter time cycle to market, have no NRE costs associated with them, can offer re-usable IP options, can be used across many platforms / PCBs to reduce inventory costs, and when using them in conjunction with a soft core embedded processor, can solve the device obsolescence problem.

References

1. *The Economist*, Bespoke chips for the common man, 12 December 2002.
2. *SOC SIP in the Fabless Supply Chain*, John Ford, Vice President of Marketing, Virtual Silicon Technology FSA Fabless Forum, September 2001.
3. *iSuppli*, ASIC Design Starts 2002: Reflecting the Past, Foreshadowing the Future, 2002.

4. *Celoxica White Paper*, Real World Experiences Designing For Mixed CPU + FPGA Systems, August 2002.
5. *Embedded Systems Reading Group Presentation*, Department of Computer Sciences, National University of Singapore, Tulika Mitra, July 2002.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
7/12/04	1.0	Initial Xilinx release.
7/21/04	1.1	Revised Fig. 1, Fig. 2; made minor text edits.