

FEATURE ARTICLE

George Novacek

Decisions, Decisions...

Choosing the Right Technology

Are you creative? Well, if you are the engineer who has to make the decision concerning a certain design, then you had better be. George discusses some of the more important aspects of decision making and shows us that, by understanding the problems and options available, we can decrease the risk and choose the right alternative.



A number of articles have been published in *Circuit Cellar* about FPGAs. I don't want to rehash what has already been so eloquently stated by Ingo Cyliax, Jan Gray <Link to past articles?>, and others. I recommend that you go online or dust off back issues of *Circuit Cellar* and read up on the technical details of working and designing with FPGA. In this article, I want to focus on a different but equally important aspect of the subject.

Suppose you are the leader of a design team who has just been handed a new project. The first and most important design decision facing you is what technology to use to deliver the new product on time, under budget, acting precisely within specifications. Making the wrong choice early on can cost you dearly later. You might finish with a product not performing as it should, it may cost too much to design and manufacture, or you may completely blow the sched-

ule and wind up too late to the market.

The bad news is that there is no cut-and-dry rule to offer in making this decision. Being creative comes along with the territory in the life of an engineer. As a team leader, an engineering manager adds more to the design creativity, like the need to make judgment calls and take certain risks. In every design process, you can hedge bets for only so long, but eventually you reach the point of no return (early in electronics design) when you must make a choice and follow through with it. The silver lining in this rather depressing scenario is that, by understanding the problems and choices available, you can mitigate the risk. In this article, I'll discuss some of the more important pieces in the decision process to help you choose the right alternative.

THE PERFORMANCE

Everything is secondary to performance. Although the interdependence of performance, schedule, and cost cannot be separated, there is usually little, if anything, that is negotiable by the time the project's been handed to you. The purpose of this article is to illustrate how to select the right technology for the job, it is not a discussion of marketing strategy. So, I assume the marketing crew has defined what they want and how much they're willing to pay for it. As far as you're concerned, the specification is cast in stone.

THE COST

Time is money, so they say, especially with the cost of engineering time these days. And because the project cost and schedule are closely related, the fundamental question boils down to deciding what to use. Is it better to use complex, sophisticated, more expensive components,

bringing the design to market quickly, or spend more time designing in order to use simpler, less expensive, common components? In your business model, you also have to include the cost of manufacturing—weighing a single FPGA, for example, against a PCB populated with several hundred discrete components, the cost of troubleshooting, testing, and product support several years down the road. One way or another, the development cost will be reflected in the selling price of the product. Completely different scenarios will be present in a consumer-oriented product, like a cellular phone, which is built by the thousands and has a lifecycle of a few years. Place this in opposition to an avionic controller with perhaps 300 units manufactured over a period of 20 years (with no re-design), or a customized one-shot deal design.

In a practical sense, life is too short to be wasted on endless studies. As a manager, you are expected to fall back on your general knowledge and experience, cut through the maze of useless data, and make your decision quickly without agonizing over it as the days turn into weeks. Anything that shortens the time to market, reduces the risk of not meeting specification requirements, and simplifies the manufacturing process is a winner in my book, even though component cost may be greater. I will consider this approach first and, unless there is a bona fide showstopper, I won't waste time reviewing other traditional methods.

There are three reasons for this sentiment. You never fully appreciate the gain from being quick, but you will definitely feel the pain from being late. Also, microelectronics prices are always on a sliding scale. A complex, expensive part today may be inexpensive by the time your design is finished, but be careful not to fall into the trap I touch on next. Finally, engineering and troubleshooting time can be expensive, difficult to predict, and always expanding beyond original expectations. With lower proportion engineering represented in the overall cost of your project, the completion estimates will be better.

THE TRAPS

Marketing hype will make you believe that any given technology is the panacea for future problems. Maybe, but probably not. Let's say that based on the performance, development, and manufacturing aspects, it appears to make sense to implement your project using a widely advertised new technology. This may sound great, but have you considered the talent (competence) you have available for the project? Have your engineers designed a product using the new technology, or do they just think it would be a neat challenge? How much are the development tools going to cost? Are you prepared to foot the bill for pioneering new technology? Even established manufacturers have been known to market new technology before it was ready, using it as a trial balloon, then dropping it when the expected interest does not materialize.

Before settling on a certain technology, you need to understand how many units will be manufactured, how long the product will be manufactured, and how long it will have to be serviced. Component obsolescence is a serious issue, especially with the most sophisticated microelectronic components having a lifecycle of only a few years. Even simple PALs, such as the ubiquitous 22V10, have been improved and modified so many times that the new batches don't always work in older designs.

There are companies doing business that track microelectronic components through their lifecycle and provide the data to the industry. TacTech of California is one of them. To give you an example of the seriousness of the obsolescence situation, in the fall of 1999, there were about 430,000 microelectronic component offerings on the market. Out of those, some 170,000 are modified, and about 40,000 are discontinued every year. That is nearly 50%! Approximately 110 microelectronic components are discontinued every day (Independence Day and Christmas included). If your product is to be manufactured for any appreciable period of time, you will not be able to forget it and move on to

better things. Instead, a significant amount of engineering effort will be spent on keeping the production line running.

Be careful before you jump on the new technology bandwagon. To start, make sure the new technology works as advertised and make sure you understand exactly what is being advertised. I'll never forget the time I had to recall a product because of an undocumented characteristic in a new and exciting component, which represented only a miniscule portion of the overall product cost. As you might expect, its manufacturer was prepared to reimburse me for nothing more than the cost of that component only. By pioneering a new concept, you are taking a risk, which may not be appropriate for a long-term product and support.

Still hearing about tunnel diodes, PUTs, and UJT's? A good example may be a line of I/O chips combined with memory, which was introduced several years ago. A great idea because it worked well and saved a lot of real estate on the PCBs. Unfortunately, it was discontinued without warning at the same time my company was putting it into production. Redesign, recertification, and one unhappy customer were the results of this undertaking, something I would not like to relive. The thousands of dollars worth of development and programming tools can be used as boat anchors now.

TRAINING

Vendors of microelectronic components run training programs for design engineers in all major cities across North America. Fees for these courses range from free to several thousand dollars, the expensive ones including development kits and hands-on workshops. Although the prices for good programs may seem steep, in my experience, it is more economical than learning on the job through trial and error. I've discovered that the best investment is to follow a training course by giving each engineer a development kit to take home. Though we may deny it, we are engineers at heart because we like to play with

cool stuff. For the low cost of the toy, the engineer will learn the new technology quickly at home, at no additional cost to your company.

TAKE THE PICK

Faced with a new project, you have essentially three ways to go—you can build the new design with discrete hardware, use a microprocessor-based design, or an FPGA.

You will always need some analog circuitry to interface with the real world, and the discrete hardware design should be kept predominantly analog. Using discrete logic should be limited to extremely simple, low-quantity, or glue logic design, or where characteristics of CMOS logic are desirable, such as 15-V operation. I can't think of many applications where I wouldn't rather use a PAL.

Amazing results can be quickly achieved with a few analog circuits in situations that would otherwise require a complex digital circuitry. If all you need is a few pieces of custom equipment (and you have an experienced analog designer on the team), this may be the best way to go. The design will be fast, understandable, and easy to test and troubleshoot. The necessary tweaking of a few units for proper performance is a small price to pay for the simplicity and quick delivery. Unfortunately, this type of design is becoming a lost art.

Whenever there is high complexity, high volume, or both, the digital approach is the route to take, even in analog-dominated areas, such as Hi-Fi. Analog design is more limited to pro-

viding analog-to-digital and vice versa interface with the outside world. Functions are being gradually taken over by digital processing, either by a microprocessor or logic circuits, which include the entire range of programmable logic from PALs to FPGAs to ASICs. Digital processing is inherently more robust than analog processing. Adjustments and tweaking can be done automatically and repeatability is hard to beat.

LOGIC PROCESSING

It makes sense to use SSI (Small Scale Integration) or MSI (Medium Scale Integration) logic devices only in small, simple designs or when other devices can't match their characteristics (see Photo 1). I prefer using PALs over discrete logic devices. There may seem to be little sense in swapping an LSI chip with a PAL programmed to perform the same function, until you realize it may provide additional benefits, such as better routing layout on the PCB or one less component on the materials list. The big players have been using ASICs, but for the majority of ordinary mortals, the heavy NRE cost, minimum purchase, and long lead time makes this option unattractive. Even the hot shots have an increasingly harder time justifying ASIC development over an FPGA.

So, what can FPGAs do for us and when does it make sense to use them instead of a microprocessor (microcontroller)? Both devices could be used with the majority of applications. Let's see what the FPGA has to offer (see Photo 2).

An FPGA is an array of logic elements you can interconnect in an infinite number of ways to create a logic processor, state machines, microprocessors, DSPs, digital filters, bus controllers, and so forth. Hard macros are used to program logic elements for obtaining basic functions similar to SSI, such as gates and flip-flops. Soft macros combine the hard macros to build more

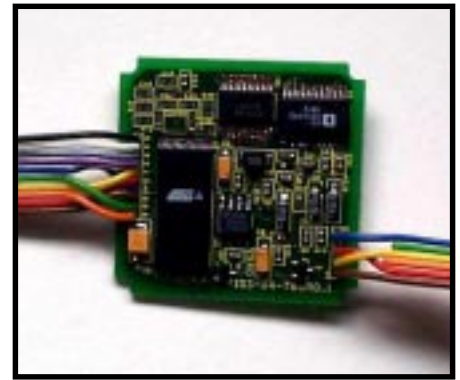


Photo 2—The same circuit redesigned with an FPGA. The PCB area has shrunk by about 90%.

complex functions, such as counters and adders. To make life easier, FPGA vendors offer libraries of IPs (intellectual properties), or cores with already developed and tested devices, such as microprocessors. These off-the-shelf IPs readily interface with most of the standard EDA (Engineering Design Automation) tools. You could literally build a single PCB with an FPGA and a few additional parts to become a different microprocessor module by simply downloading a different personality IP.

Because most FPGA processing can be done in parallel, an FPGA-based design will run at speeds limited by the internal propagation delay only. A microprocessor or microcontroller, on the other hand, executes instructions sequentially. There are many real-time processing tasks where even the fastest processors can't deliver adequate performance, so the FPGA is a logical, practical choice.

Now that I have presented the FPGA as nearly capable of walking on water, why would you want to use a microprocessor at all? The fact is that the FPGA is making serious inroads into the microprocessor's exclusive territory. However, the FPGA-based microprocessor does not yet measure up to the performance of the modern, high-end micros. If the primary task of the design is many intelligent interfaces, data processing, and manipulation or programmable mathematical functions, a micro is still more efficient. The Pentium in the PC can't be replaced by an FPGA any time soon, but given the speed of progress, you never know.

When it comes to embedded con-



Photo 1—Here you can see a partial view of an autorigging module implemented with traditional SSI and MSI logic circuits. Components are mounted on both sides of this SMT circuit board. How would you like to troubleshoot this circuit?

trollers, the microprocessor's strengths and weaknesses go hand in hand where software is concerned. Let's take a closer look.

SOFTWARE

Countless horror stories have given software a bad reputation, so much so that the development efforts today (especially in terms of product safety) are devoted to documenting and testing. In other words, proving the absence of bugs. To be fair, this is not merely the fault of undisciplined or unscrupulous software developers and publishers. I remember the marketing stories that accompanied the first microprocessors, practically guaranteeing how inexpensive product development and manufacturing will be in the days to come. This imaginary pot of gold stuck in our minds so that even today few design managers have come to terms with the true cost of software development. Then, having quoted unrealistic costs, they are forced to cut corners and release software before it is ready.

The general rule of thumb is that one line of code in C will take about three hours of engineering time. Ten thousand lines of code in an embedded controller at the industry rate will cost you a cool quarter of a million. You can calculate the exorbitant cost of 30,000 hours represented in product delivery if you have four software designers working 2,000 hours per year. In commercial projects not requiring the stringent documentation and testing needed for safety critical software, one to two hours per line is realistic, but still, go ahead and do the math! Many recently-introduced EDA tools automate documentation, traceability, and testing so you can see significant improvements in productivity, but the fact remains that software development is long, expensive, and will never reach the level originally envisioned and promised by marketeers.

There are people who may argue that they "whipped up" some code in only a few short hours. To them I will repeat that, even if you get away with cutting a few corners now, by the time you factor in the failures in the field,

unhappy customers, and next-to impossible code maintenance, you would have been better off doing the job properly in the first place.

In contrast, I have always likened the design of FPGAs or PALs to laying out a PCB. Although the routing is done with the help of software, the result is a fully testable hardware. This eliminates a large proportion of the documentation and testing otherwise required for software, which is an incentive to design with the FPGAs. For safety critical applications, the certifying authorities are getting wise to the growing use of the FPGAs, but their inherently faster and self-documenting development could not be inflated to the required level for certifying software.

WHAT'S NEXT?

The undisputed flexibility, quick turn-around time, almost instant prototyping, and drastic reduction in package count make the FPGA the technology of choice for the majority of future products. This trend is reinforced by the introduction of the SoC (System-on-Chip) initiative, as well as the ever-growing libraries of IPs. I see a definite trend towards nearly exclusive use of FPGAs in embedded systems, where they can deliver efficient processing, short development time, instant prototyping, and easy in-field modifications. Anyone who has had to deal with long cycle times in closed-loop control systems will appreciate the speed of FPGAs.

With the help of FPGAs, we are getting closer to the dream of a universal hardware platform. As long as there is sufficient I/O capability, the PCB can be reprogrammed for any desired function. That means benefits like less hardware development for the ever-present expensive goofs and costly environmental testing, higher manufacturing volume, automated testing, less troubleshooting, better reliability, and lower cost. To put it simply, heaven on earth. Painful software development and certification will be left for applications software. Discrete design? That, my friends, is a dying art...but it was fun. ☑

George Novacek has 30 years of experience in circuit design and embedded controllers. He is currently the general manager of Messier-Dowty Electronics, a division of Messier-Dowty International, the world's largest manufacturer of landing-gear systems. You may reach him at gnovacek@nexicom.net.

SOURCE

Microelectronic components

TacTech

(714) 974-7676

Fax: (714) 283-3213

www.tactech.com

REFERENCES

- [1] Fraunhofer-Gesellschaft, "Fraunhofer IIS-A-Layer-3 Info," <http://www.iis.fhg.de/amm/techinf/layer3/index.html>, 1999.
- [2] Fraunhofer-Gesellschaft, "Fraunhofer IIS-A-Layer-3 FAQ," <http://www.iis.fhg.de/amm/techinf/layer3/layer3faq/index.html>, 1999.
- [3] T. Sporer, K. Brandenburg, B. Edler, "The Use of Multirate Filter Banks for Coding High Quality Digital Audio," sixth European Signal Processing Conference (EUSIPCO), Amsterdam, 1992.
- [4] "The Discrete Cosine Transform," <http://dynamo.ecn.purdue.edu/~ace/jpeg-tut/jpgdct1.html>, 1999.
- [5] M. Fattouche, "Free and Legal MP3 Music," University of Calgary ENEL 571, Lectures Notes, <http://www.mpeg.org/MPEG/mp3.html#faqs>, 1999.
- [6] RIAA Press Release, "Music and Technology Companies Join to Develop Means to Protect Copyrighted Music," <http://www.techlawjournal.com/intelpro/19981216.htm>, 1999.

Circuit Cellar, the Magazine for Computer Applications. Reprinted by permission. For subscription information, call (860) 875-2199, subscribe@circuitcellar.com or www.circuitcellar.com/subscribe.htm.