

## LESSONS FROM THE TRENCHES

George Martin

# What's the Logic Behind the Design?

## Working with Programmable Logic

Last month, George gave us some great new product ideas. And, while watching CSPAN one evening, he got another great idea, which in turn got him thinking about logic devices. Join him, this month, as he takes us through the process of programmable logic.



was watching CSPAN one evening and Scott McNealy of Sun

Microsystems was on talking about Microsoft's new .NET initiative. That's an interesting topic by itself, but the part I found exciting was his list on new products. Remember last month's article?

The new product that caught my attention was a unit that monitors the tire pressure in your car. Scott, of course, has it Java/web-enabled so that it reminds you when tire pressure is low and then sends an e-mail to the manufacturer. He suggested that a rollover sensor could work the same way and call for help whenever the car was upside down. If you think about it, all the necessary components are already available for such a system. It just needs to be implemented. There's another good product idea for you.

If you've ever had to program logic devices, you've probably used a text-based programming language such as PALASM or ABEL. I was first introduced to programmable

logic with the Advanced Micro (AMD) MACH devices. In the late '80s, AMD had a traveling presentation that was given by the actual designers of the compiler. That presentation added to my knowledge base and has kept me up to date since.

Those early PLD devices started as 20-pin ICs. Using one pin for power and one for ground, that left 18 pins for inputs and outputs. The simple 16L8 device had 10 inputs, six input/outputs, and two output pins. The latched version (16R8) had one clock pin, one output enable pin, eight input pins, and eight output pins. I did a lot with these devices, filling them to 98% of capacity. These devices had more pins and more logic, as well as custom organizations for specific applications such as state machines.

The next big step along the integration curve was to place four copies of the eight-register devices in a single package. Soon after, devices similar to AMD's MACH 110 started appearing. The MACH 110 was a 44-pin PLCC package with two clock inputs, five general-purpose inputs, and four banks of eight registers each for input/output. Also offered was the MACH 210, which had an additional four banks of eight registers. These additional registers could be used as buried logic devices and did not appear at the pins. These were great for long counters that had no need to

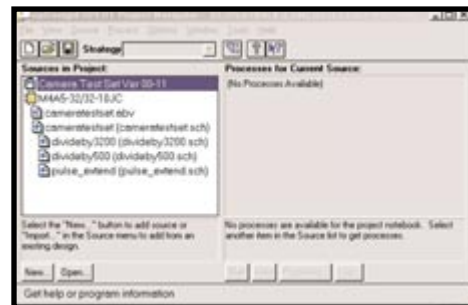


Photo 1—The top-level view of the project navigator is always visible.

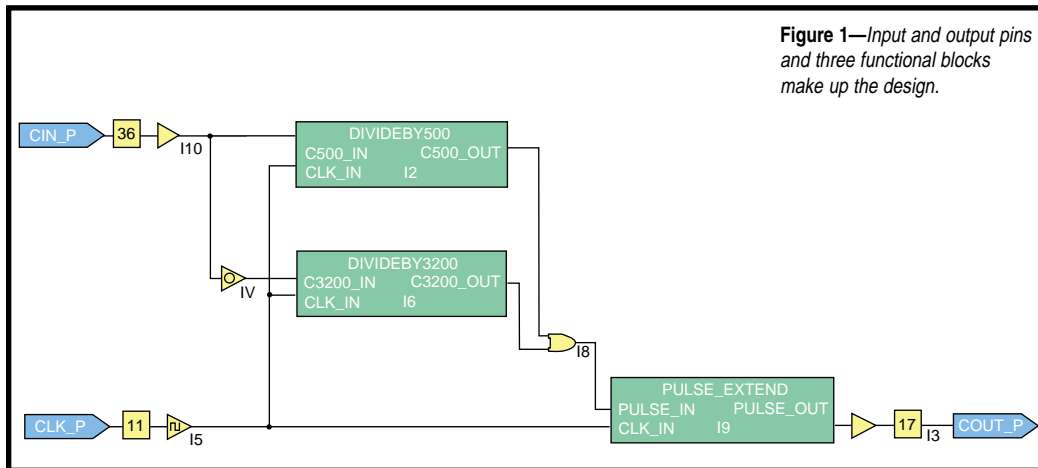


Figure 1—Input and output pins and three functional blocks make up the design.

report intermediate results or status. However, it became difficult to write all the code and run simulations on a complicated design—too many pages and too difficult to follow.

Well, it's over 20 years later and times are changing. AMD spun the PLD business into a separate company named Vantis. Vantis was then purchased by Lattice Semiconductor.

## THE SEARCH BEGINS

I recently needed to design a simple test set. I needed to use the latest devices, and that meant the latest software. Searching the Internet, I ended up on the Lattice Semiconductor home page. Working my way to the free software section, I found ispDesignEXPERT Starter. It's a big download, but it's worth it, as you will see.

Manuals and tutorials come with the product. You can print them out or follow an online slide show as if in class. This is a big help because, in about a week, you're ready to tackle a new design.

ispDesignEXPERT is an integrated environment that makes the PLD design problem a whole lot easier. The PLD portion of the test set I was designing needed to receive a clock and divide that clock by either 500 or 3200. The divide selector was an input pin, and the divided signal was an output pin. So, I was looking at a good first project.

## NAVIGATING

Photo 1 shows the top-level view of the project navigator, which is always

visible. I've named the project Camera Test Set V.00-11. In the directory, I control each revision by creating a new directory for that revision.

The next entry is the M4A5-32/32-10JC. That's the specific device I targeted my application to. You can try different devices and see if they fit and how the design works. If a design fits but is 98% full, you may not want to release that into production. If you select this line in the project sources window, you'll see the steps to compile, fit, and simulate the design.

As you look through the processes shown in Photo 2, you get a feel of the organization and power of this system. Many of these features come from the high-end work station design packages. The maintenance alone for these high-end packages is over \$100,000 per year.

Double clicking on the Fit Design process will do just that. A green check indicates successful completion, and a red "x" indicate a warning or error in the process. If you're accustomed to PLD design, then processes like Constraints Editor will have some meaning to you.

The next file cameratestset.abv is a file automatically created by the software. Do not edit this file because it will be overwritten.

The next file, cameratestset.sch, is the top-level design file. Figure 1 shows the input and output pins and the three functional blocks that make up the design. The clock input is pin 11, the rate selector is pin 36, and the output is pin 17. The inverter and OR gate have

instance numbers associated with them. The inverter is I7, and the OR gate is I8. I've also defined a divide by 500, divide by 3200, and a Pulse\_Extender functional block.

Figure 2 shows the details of the divide by 500 implementation. The program would have permitted this to be an ABEL text page as well as a schematic diagram. A test called Latched Reset

is added free form, and I think it adds a lot to the understanding of the design. The basic building blocks consisted of a toggle (T) flip flop with a reset input. But, this reset is asynchronous.

## COUNTING THE WAYS

In the past, I've written about PLD design approaches. And, I've suggested that you'd better use synchronous design if you want to survive. Well, as I was simulating this design, I counted to 499 and a reset signal appeared and started to reset the flip-flops. As soon as the first flip flop was reset, the reset signal went away and the reset process was stopped. So, I added a flip-flop I24 to make the reset synchronous.

This design counts from 0 to 498. Note that that's 499 states. Then I22, I25, and I21 decode the state value of 498. The output of I21 goes high and is latched into I24. That latched value is then used to reset all the toggle flip-flops. This reset signal is also the output of the divide by 500 circuit. The output pulse is one clock wide.

The gates I22, I25, and I21 are used

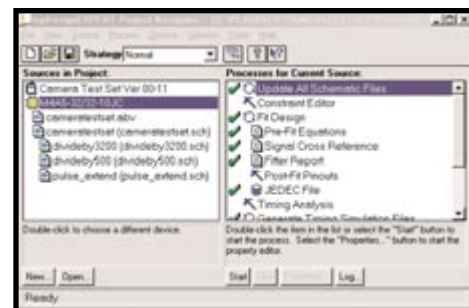


Photo 2—Here you can see the processes for the device.

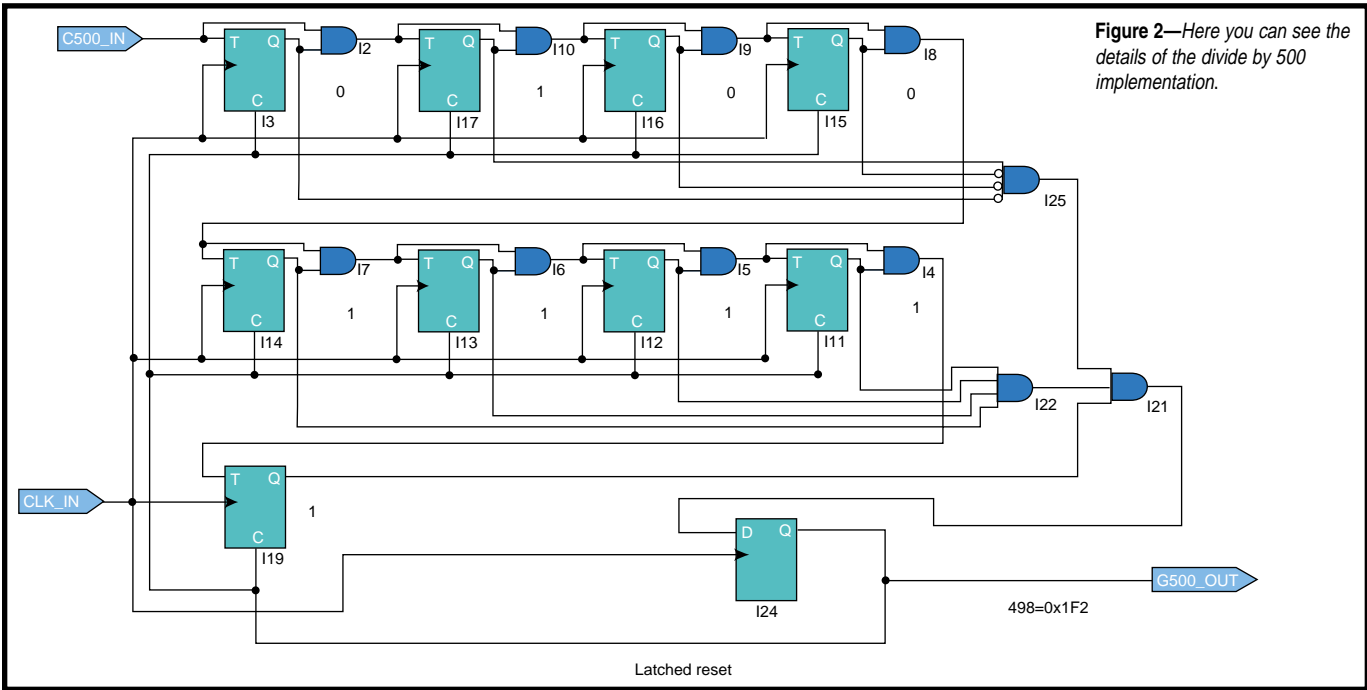


Figure 2—Here you can see the details of the divide by 500 implementation.

to implement the function:

$$\text{Reset} = *Q[0] \times Q[1] \times *Q[2] \times *Q[3] \times Q[4] \times Q[5] \times Q[6] \times Q[7] \times Q[8].$$

Be certain that the structure of the PDL supports such equations. This is a simple example, and I'm sure the MACH devices can handle a nine-input function. The schematic elements available are limited to four input devices, so you need to create larger input function out of the building blocks. Just don't get carried away, or it'll be impossible to fit the results into the device.

I needed to stretch the output of divider circuitry. Figure 3 shows the simple piece of logic that captures the input signal. If it's a one, then it starts a 3-bit counter, which keeps counting until state 0x000 is reached, then it stops counting.

### SIMULATION

The final step in the design process is simulation. Both functional and time simulations are supported. Photo 3 shows the output pulses at the divide by 500 setting. As you can see, the output pulses are 500,000 timing units apart. I set the clock to have 500 timing units high and 500 timing units low for a period of 1000 timing units. So, output pulses of 500,000 represent a divide by 500.

This is, of course, a brief overview of the software. It's free and it's valuable. It sure beats the text version. And, building a design base is straightforward with this software. ☑

*George Martin began his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and cofounded a design and manufacturing firm. Typical systems that George designs include servo-*

*motion control, graphical input and output, data acquisition, and remote control. George is a charter member of the Ciarcia Design Works Team and most recently, he's been working on the people-tracking system for Bill*

### SOFTWARE

The ispDesignEXPERT Starter software can be downloaded from the Lattice Semiconductor web site at [http://www.latticesemi.com/products/devtools/software/des-expert\\_pc8\\_starter.cfm](http://www.latticesemi.com/products/devtools/software/des-expert_pc8_starter.cfm).

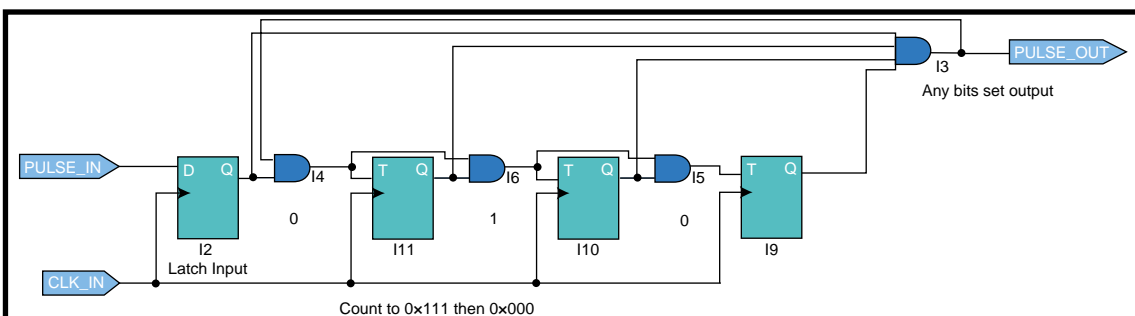
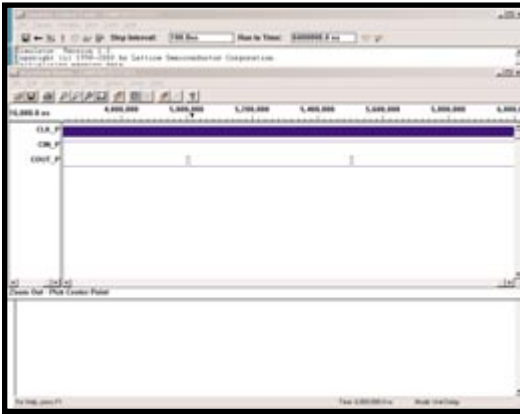


Figure 3—The simple piece of logic that captures the input signal can be seen here.



**Photo 3**—This screen shows the output pulses at the divide by 500 setting.

## SOURCE

### **PALASM**

Advanced Micro Devices, Inc.  
(800) 538-8450  
[www.amd.com](http://www.amd.com)

### **ABEL**

Xilinx  
(408) 559-7778  
Fax: (408) 559-7114  
[www.xilinx.com](http://www.xilinx.com)  
<http://www.xilinx.com/products/software/mincsynario/ms-infosite.htm>