Embedded Networking with the iMCU W7100, p. 14 • Extend the I²C Bus, p. 64

www.circuitcellar.com

CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

#233 December 2009

PROGRAMMABLE LOGIC

Retrocomputing with Programmable Logic

Microprogramming with FPGAs

Processing Memory Failures

Digital Modulation Theory

6LoWPAN Explained

0 74470 75349 0 12

$5.95 U.S. ($6.95 Canada)

INSIDE BONUS ISSUE 233

BONUS ARTICLE:

Page 1: The Evolution of Rabbits
Five Generations of Rabbit Microprocessors
by Monte Dalrymple

ARTICLES THAT APPEARED IN SUBSCRIBER COPIES OF ISSUE 233:

- **iMCU W7100**
  Embedded Networking Made Simple
- **Retrocomputing on an FPGA**
  Reconstruct an '80s-Era Home Computer with Programmable Logic
- **Building Microprogrammed Machines with FPGAs**
- **ABOVE THE GROUND PLANE**
  Memories Are Not Forever
- **THE DARKER SIDE**
  Digital Modulations Demystified
- **SILICON UPDATE**
  IP Unplugged
- **FROM THE BENCH**
  Extend and Isolate the I^2C Bus

To purchase any of these subscriber-only articles in PDF format, visit http://www.circuitcellar.com/magazine/233.html

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

Subscribe to Circuit Cellar's Print or Digital edition today:
Visit www.circuitcellar.com/DP

WELCOME…

WHAT YOU SEE HERE TODAY IS A PRESENTATION OF BONUS MATERIAL THAT FIRST APPEARED IN CIRCUIT CELLAR'S DECEMBER 233 DIGITAL PLUS EDITION.

THE DIGITAL VERSION OF THE MAGAZINE ALLOWS CIRCUIT CELLAR TO PUBLISH MORE CONTENT EACH MONTH, INCLUDING ARTICLES OF LENGTHS PROHIBITIVE TO PRINT PUBLISHING.

THE PRINT MAGAZINE ISN'T GOING AWAY (SEE TOC TO LEFT FOR THE KINDS OF ARTICLES THAT CONTINUE TO APPEAR IN THE PRINT VERSION). BUT THE BONUS DIGITAL EDITION YOU SEE HERE OFFERS AN EXCELLENT WAY FOR PRINT SUBSCRIBERS TO SECURE EVEN MORE CONTENT. PLEASE ENJOY THIS COURTESY COPY!

Note: *If you would like e-mail notification when bonus content becomes available, be sure to subscribe to Circuit Cellar's e-mail newsletter for regular announcements.*

-Subscribe to Newsletter-

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

BONUS ARTICLE

by Monte Dalrymple

The Evolution of Rabbits

Five Generations of Rabbit Microprocessors

How do IC designers deal with changing technology? To answer that question, let's review the evolution of a processor family over time.

In 1997, I was approached with the idea of developing a proprietary alternative to the Zilog Z180 microprocessor. At the time, the Z180 was getting long in the tooth and later Zilog microprocessors, some of which I had worked on, weren't sufficiently compatible for the folks at Z-World (now a part of Rabbit Semiconductor).

At the start of the project, I don't think that anyone expected that we would end up doing multiple generations of the design. But part of the job of a CPU designer is to plan for the future by avoiding design decisions that might come back to haunt the unwary. The goal of this article is to detail the evolution of Rabbit microprocessors over five generations, while dealing with changes in process technology, packaging technology, and the feature set.

DEALING WITH MOORE'S LAW

Moore's Law states that integrated circuit complexity doubles about every 18 months. Dealing with this moving target can be very challenging. For example, if the design cycle time from concept to tape-out is a little over two years, you need to start the project based on assumptions that won't be economically viable until the project is nearly complete. In addition, any delay in the project means that you are not taking full advantage of technology.

These facts give engineers headaches, but they also mean that the people who worry about development costs and return on investments (i.e., the bean counters) have to be technically savvy to make investment decisions. Aggressive technology companies count on Moore's Law for their product development, but newcomers like Z-World are forced to be very conservative with their development money.

This fact is evident when you look at the information in Table 1, which illustrates the march of technology over five generations of microprocessors. As the table shows, we were very conservative with the first two generations, and didn't aggressively push the technology until the latest generation. Table 2 details how the features have changed over

Feature	Rabbit 2000	Rabbit 3000	Rabbit 4000	Rabbit 5000	Rabbit 6000
Voltage (IO/core)	5.0/5.0	3.3/3.3	3.3/1.8	3.3/1.8	3.3/1.2
Clock speed	30 MHz	55 MHz	60 MHz	100 MHz	200 MHz
Package pins	100	128	128	289 or 196	292 or 233
Technology	0.6-µm gate array	0.35-µm gate array	180-nm std cell	180-nm std cell	90-nm std cell
Gate count	19K	31K	161K	540K	760K
Embedded RAM	none	none	256	141 KB	177 KB
Executable RAM	none	none	none	1-MB SRAM	8-MB DRAM 256-KB SRAM

Table 1—The march of technology is clear in each row of the table. While we squeezed every gate out of the Rabbit 2000, in the 6000 the logic that we actually designed was only a small fraction of the total.

Feature	Rabbit 2000	Rabbit 3000	Rabbit 4000	Rabbit 5000	Rabbit 6000
Processors	1 CPU	1 CPU	1 CPU	2 CPUs 1 DSP	4 CPUs 2 DSPs
Parallel Ports	5	7	5	6	8
Serial Ports	4	6	6 (plus BRG)	6 (plus BRG)	7 (plus BRG)
Timers	5× 8-bit	10 × 8-bit	10 × 8-bit	10 × 8-bit	13 × 8-bit
	2× 10-bit	2 × 10-bit	2 × 10-bit	2 × 10-bit	2 × 10-bit
		1× 16-bit	1 × 16-bit	1 × 16-bit	1 × 16-bit
Other Functions		Capture, PWM, Quadrature	Capture, PWM, Quadrature	Capture, PWM, Quadrature	Capture, PWM, Quadrature, 2x FIM
Network	none	none	10Base-T	10/100, Wi-Fi	10/100, Wi-Fi, USB

Table 2—The feature set grew with each generation. With the 6000, most of the complexity came from integrating functional blocks designed by someone else. (BRG stands for "baud rate generator.")

time. Notice the drastic changes between the first generation and the fifth generation.

THE RABBIT 2000

To understand the Rabbit 2000, you have to start with the technology that was used for its implementation: a gate array. Gate arrays come in discrete sizes, usually varying by a factor of about 1.5 for the number of gates available. They are also limited as to the number of pins available, with a fixed number of pads on the chip and only two or three package pin counts available for each gate array size.

While these limitations might seem excessive, they result in significant cost savings because you only have to pay for the masks used to wire up the transistors rather than a complete set of masks. So, instead of paying for 20 or more masks, you only have to pay for half a dozen.

The big problem is choosing a target gate array for the design. In the case of the Rabbit 2000, the primary consideration was the package and pin count. Z-World wanted a 100-pin PQFP package, and that immediately limited the gate array size to 25,000 gates.

With this hard limit in place, I started the project. Z-World had a wish-list of features for the CPU, including a few new instructions and a list of Z180 instructions that were not needed. They also had a list of peripherals and features to reduce board costs.

At the time pipelines and single-cycle execution were all the rage, but careful analysis revealed that this wasn't the way to go for this design. The problem with pipelines is that they require more logic, and single-cycle execution means that you don't have a lot of clock edges to use for signals when talking to external memory.

Since one of the objectives was to minimize board cost, with direct connection to standard memories, we settled on a two-clock basic machine cycle. This basic timing has been used for all five generations, and as I'll explain later, has provided a number of advantages down the road.

With the instruction set and basic timing chosen, I started implementing the CPU. But the peripherals were a different matter. Many engineers will want to dive right in and start designing. After all, that's the fun part of engineering. But long experience has taught me that it's better to spend time in the beginning clearly defining the programming interface and timing for the peripherals.

So, while I was designing the CPU in parallel I was writing what would later become the user manual for the peripherals. Having a complete user manual allowed the software folks to review and comment on the register definitions and actually start coding drivers before the hardware even existed.

At the same time, the hardware engineers at Z-World were designing a board containing a large FPGA to verify the design before we released it to the fab. Z-World had initially wanted to do the design using schematics, but it didn't take much to convince them that a hardware description language was the only realistic way to go. Using Verilog HDL allowed us to target the design to FPGAs from two different vendors as well as the final gate array with only a few differences in the source code.

The one disadvantage of using a hardware description language is that it's hard to get a feel for how many gates you're using until the project is well under way. In fact, the first synthesis result exceeded the gate limit slightly. Since we weren't sure how well the autorouter would do in placing the design into the gate array, this caused no small amount of consternation.

After looking carefully at the synthesis results, we decided on a few features to remove. Some of the features that were removed would create challenges that would persist for several generations.

The most painful change was to remove the ability to read back the contents of the peripheral control registers. In my previous experience designing peripheral devices, this was a feature that was always requested by customers, and it also makes simulation and testing much easier. But Z-World, as the authors of most of the software that would be using the design, felt that the feature wasn't really necessary.

Another change that would have implications in later generations was the addressing for the internal peripherals. Rather than using the entire 16 bits of I/O address, the internal peripherals in the Rabbit 2000 only decode the lower eight bits of the I/O address.

I had originally specified all of the parallel ports as completely programmable as far as data direction; but since many of these pins also provided access to the serial ports, we ended up restricting some of the ports to a single direction.

Finally, changes were made in the serial ports, restricting two ports to async-only and removing features like dedicated baud-rate generators. Most people think that this is why parity was not included in the serial ports, but they are

wrong. Norm Rogers, the president of Z-World, maintained that parity was obsolete, and had no place in the design. He even insisted that the parity flag operation that was part of the Z180 instruction set be removed. Needless to say, customers did not agree, and parity had to be implemented crudely in software.

As the design neared completion it became apparent that we might have a hit on our hands. The software was coming together, and customer feedback was already very positive. To create a "brand" Z-World went looking for a name for the processor. Note that 1999 was the year of the rabbit in the Chinese Lunar Calendar and that's where the Rabbit Semiconductor name came from. Since the design would be introduced in 2000, someone came up with the moniker Rabbit 2000.

THE RABBIT 3000

The Rabbit 2000 started selling very quickly, and just as quickly we started getting feedback from customers about features that they wanted. At the same time, software started talking about an operating system, and the hardware group gave feedback about the board designs.

All of this feedback led to the start of the Rabbit 3000 project. As before, the first decision was pin count and package. This time the choice was 128 pins and TQFP. The problem with this choice was the number of gates available in the 0.6-µm technology of the 2000. There just weren't enough gates available to make this a reasonable next step.

The end result was a change to the next available technology, which was 0.35 µm. This gave a significant boost in the number of gates available, but had the downside of requiring a 3.3-V supply.

The feedback from software resulted in adding 14 new instructions to the instruction set. With the methodology I have developed, over many years of designing CPUs, this was a simple change. More complex was adding support for an operating system.

This required fundamental changes in the guts of the processor to support separate System and User modes of operation. In addition, the 8 bits of internal I/O address space was nearly full and there was no room for many of the new registers required for these features. I was able to make the increased internal I/O address space mostly backwards-compatible. And although the System/User mode has continued in later generations, the software support for the feature never materialized in any significant way.

The customer feedback resulted in the addition of more parallel ports, and more serial ports. The six serial ports on the 3000 were the most of any 8-bit microprocessor, and two of the ports added full HDLC capability.

Customers also wanted more support for motion control applications, which led to the addition of pulse-width modulators, input capture channels, and quadrature decoders. Even though we had more gates available—and by this time everyone was complaining about write-only peripheral registers—no changes were made in this regard. And there was still no parity in the serial ports.

A number of other new features were aimed at reducing the power consumption of the design. Internally, I changed all of the peripheral control registers to use gated clocks and latches instead of clock enables and flip-flops. Normally, gated clocks are an absolute no-no in digital design, and every time we go to fabricate a new generation the fab will complain loudly. But the two clock-cycle machine cycle is ideal for guaranteeing setup and hold times around the gated clock, and we've never had a problem with this technique.

Careful characterization of the Rabbit 2000 had revealed that the slowest path in the design involved the address translation in the MMU. I came up with an alternate implementation that used about four times as many gates but was about four times as fast. After the 3000 came out and proved the design, it was fed back into a revision of the 2000, along with the new spread-spectrum clock generator.

THE RABBIT 4000

In some ways the Rabbit 4000 is an anomaly, mostly because of the package that was selected by Z-World. At the time that the project was started, a majority of the Rabbit-based boards included a 10Base-T network port, and Z-World wanted to bring this functionality into the next generation. But keeping the 128-pin package meant some serious compromises. And the estimated gate count dictated that we move to a smaller process geometry, with split power supplies for the core and the I/O.

This meant removing the two parallel ports that we had added for the 3000 to make room for the network connections and new power pins. In retrospect, this was a mistake, because this meant that all of the other peripherals had to share fewer pins. So, not all of the peripherals could actually be used at the same time.

At the same time, Z-World wanted to provide the option of using 16-bit memories, potentially taking away another nine pins (eight for data and one for the byte/word selector). The hardware guys and I argued in vain for more pins. But at least we were finally able to incorporate parity (without telling Norm) and dedicated baud rate generators into the serial ports.

Although 10Base-T (and 10/100) cores were available for purchase, the Z-World philosophy was to design it in-house to maintain control. So, I was introduced to the world of IEEE standards, and spent about six months designing to that specification.

The result is actually fairly unique. Norm Rogers wanted to avoid having to use an external physical interface (PHY), and instead use some simple external components to take care of the analog requirements. So the design is a hybrid combination of the Media Access Controller (MAC) and PHY.

Rather than the typical large buffer for the network port, holding a full frame of data, Z-World asked me to analyze the requirements to use small FIFOs and add a new DMA capability to the design. Adding DMA to the design was another major task, because in the very beginning, with the Rabbit 2000, the direction was that there would never be a need for DMA.

The network port and eight channels of DMA created an issue with the interrupt vectors. Backwards-compatibility was not possible for the interrupt vector table. But despite repeated warnings about the changes to the interrupt vectors, the software folks were still surprised by the change when the chip came out.

The Rabbit 4000 marked the first major architectural upgrade to the CPU, with new registers and a number of new instructions. Code analysis had revealed that there weren't really enough CPU registers to hold pointer addresses. So the software folks wanted to add three or four 24-bit pointer registers that would hold physical addresses.

Besides being an architectural wart, this request was clearly short-sighted. In the end we were able to argue for a total of eight new 32-bit registers that could be used for data, logical addresses, or physical addresses. These registers would eventually allow the Rabbit CPU to move to full support for 32-bit operations.

The new instructions to support the new registers eventually numbered more than 200, and rather than add them in a backwards-compatible fashion Z-World required a mode bit to control access to the most important new instructions. I personally don't like mode bits, but then I don't write software for a living. The rationale was improved code density because backwards-compatibility would have meant larger opcodes.

Remember the write-only peripheral control registers? The software folks had ended up keeping copies of the registers in a table in external memory, and using those contents when modifying register contents. This required several instructions, so they wanted a new complex instruction that would read memory, modify the bits under a mask, and write the results back to memory and to the peripheral control register. I implemented the new instruction; but like the System/User features in the 3000, the instruction was only used three times in the software.

The main reason that happened was that we finally made all of the peripheral control registers readable. When we sent a trial netlist to the vendor, they came back with the information that the size of the chip was limited by the number of pads and we had plenty of room for more gates. In a quick scramble, I added in as many features as possible in a short time.

The Rabbit 4000 had to leave the gate array technology because of the number of gates relative to the number of pins, but we drastically underestimated how much better the packing density was. In the end the logic of the 4000 required less than one third of the area available for gates, leaving lots of blank space on the chip.

THE RABBIT 5000

Just before we sent the Rabbit 4000 to the fab, Z-World was bought by a much larger company, Digi International. With this ownership change came a change in philosophy relative to design. Where Z-World had always eschewed using externally supplied intellectual property (IP), Digi actually preferred to buy rather than design from scratch. In addition, they didn't care much about pin count, preferring BGA packages to surface-mount with leads. This took some getting used to.

Although the Rabbit 5000 would contain no additions to the instruction set, there was major work to be done inside the CPU. The 16-bit bus option in the 4000 used a separate prefetch mechanism that merely buffered instruction bytes. Data reads and writes were still 8 bits.

The goal in the 4000 was primarily to allow the use of 16-bit memories, rather than provide a performance improvement. But with this generation we needed to significantly improve the performance of the CPU to support new network connectivity. The end result was that I completely reworked the instruction timing to make use of 16 bits at a time, for both instructions and data.

At the same time, I revisited the MMU change that I made in the 3000. It turned out that even with the new MMU design this path was still the limiting factor as far as clock cycle time by a significant margin. Modifying the time allotted to this operation to two full clock cycles rather than the original one clock cycle allowed the processor clock frequency to nearly double.

Even though 10Base-T provides sufficient bandwidth for the types of applications that use Rabbit microprocessors, Product Marketing wanted 100Base-T. So the Rabbit 5000 uses a third-party 10/100 MAC and an external PHY. We also added back one of the parallel ports that were lost in the 4000.

But the biggest addition to the Rabbit 5000 was a Wi-Fi interface and the associated A/D and D/A converters. The design was internally developed by Digi, for an FPGA, so I had to port it to the new technology. Verilog HDL made this port fairly straightforward, basically just replacing the FPGA-specific RAM blocks with an ASIC equivalent.

The port wasn't without complications though, because the design took advantage of a RAM feature that is specific to an FPGA. The Wi-Fi designer forgot to mention that he used the "write-before-read" feature that isn't available in normal memories. It took a fair amount of simulation time to track down the problem, and in the end we ended up having to run those memories at double the clock speed to create the required memory behavior.

The Wi-Fi interface uses a lot of gates (it has an embedded CPU plus an embedded DSP) and requires a lot of pins, but we still had space available on the chip. Rather than letting it go to waste, as we had in the 4000, we added a pair of 64K × 8 static RAMs. Unfortunately, this is less than the amount of RAM that most Rabbit-based SBCs use, but something is better than nothing.

THE RABBIT 6000

Shortly before the Rabbit 5000 went to the fab, the software folks finally got around to writing software that used the new instructions and registers in the 4000 CPU. I had included some basic 32-bit operations for the new registers, but they finally realized how much they could use those new 32-bit pointer registers, if only the instruction set provided a full complement of 32-bit operations. They also wanted more support for stack-relative addressing and

more special instructions to speed up encryption and decryption. At the same time, the hardware folks clamored for more memory and an on-chip 10/100 PHY. Product marketing folks chimed in requesting higher clock speeds, a pair of the Digi-developed satellite processor modules, and USB. Thus the Rabbit 6000 was born.

All of these new features clearly required changing to a new technology because both the 10/100 PHY and the memory are very large. In fact, the 10/100 PHY, which has an internal DSP, requires more area than all of the logic in the CPU and peripherals combined. It also consumes a significant amount of power.

In the end, we added almost 200 new instructions, and they turned the Rabbit 6000 into a 32-bit machine internally. We also added a pair of parallel ports, increasing the total to eight, and upgraded the I/O capabilities to support 16-bit external peripherals.

The only way to increase the on-chip memory to the requested level was to use dynamic RAM with the attendant memory refresh cycles. This memory supports an access every clock cycle, but remember that the Rabbit CPU is at its core a two-clock machine. So the folks at Digi—being familiar with single cycle machines like the ARM—suggested a way to take advantage of the available clock cycle. This involved using those unused clock cycles to do DMA transfers.

This type of operation is fundamentally at odds with the normal DMA operation, so I ended up designing a separate DMA engine for this feature, hidden behind a common control register interface. To the programmer, it's just DMA, but the logic automatically uses the cycle-steal engine when both source and destination are on-chip. This cycle-steal operation requires dedicated busses for the peripherals that can operate this fast, leading to half a dozen dedicated data busses on the chip.

The dynamic RAM caused a couple of hiccups during the design. The datasheet that we used specified a one clock latency for read cycles. This fit perfectly with the two-clock CPU machine cycle and interleaved DMA transfers. Unfortunately, after all of the design work was done, the vendor revised the specification, to a two-clock cycle latency! This hurt doubly, because it meant a guaranteed wait state for every CPU access, and only two out of every three clock cycles useable even when the cycle-steal DMA is running. The second problem arose when we got a test chip. We always wondered why the vendor was so intent on running a test chip, because all of the IP that we were using was supposed to be silicon-proven. But when we got the test chips and tried to use the dynamic RAM it worked erratically for no apparent reason.

Fortunately, I had included a test mode that brought the internal address and data busses out to pins. One look at the logic analyzer trace revealed that the dynamic RAM was changing the output data on the wrong edge of the clock, which under certain circumstances meant an incorrect instruction was fed to the CPU. So much for silicon-proven IP.

The Rabbit 6000 is truly a System-on-Chip (SoC), containing everything necessary for a computer except for the power supply and connectors. The Rabbit processor is surrounded by three other CPUs and a pair of DSPs. Of course, one of the processors and both DSPs are deeply embedded and are not really accessible to the user, but the two remaining CPUs are self-contained satellite processors.

These satellite processors—called Flexible Interface Modules (FIMs)—are PIC clones with dedicated program and data memories that are downloaded from the main Rabbit processor. Running completely independently, they communicate via mailboxes with the main CPU and allow for the implementation of higher-level protocols such as CAN.

IC PROGRESS

As I said at the beginning of this article, I don't think anyone ever expected that there would be five generations of Rabbit microprocessors. But I find it fascinating to compare the first generation to the fifth generation. The design went from 76,000 transistors to over 15 million, and from 30 to 200 MHz. Along the way, the instruction set more than doubled, but some of the Verilog modules weren't touched after the first version.

But perhaps the biggest change was the development cost, as the cost of the masks for the Rabbit 6000 was more than the entire development budget of the Rabbit 2000. Such is the progress of integrated circuit technology.

Author's Note: I'd like to thank Norm Rogers, Pedram Abolgasem, Lynn Wood, and Steve Hardy at Rabbit Semiconductor, and also Jeff Parker and Brad Hollister at Digi International.

Monte Dalrymple (monted@systemyde.com) has been designing integrated circuits for over 30 years. He holds a BSEE and an MSEE from the University of California at Berkeley and has 15 patents. He is the designer of all five generations of Rabbit microprocessors. Not limited to things digital, Monte holds both amateur and commercial radio licenses.