NEWT Microprocessor and 41CL Progress Report

As I stated in my presentation at last years conference, this project started as a demonstration that moving a CPU design to a new technology doesn't need to be difficult or expensive. As a "hobby" project, it hasn't received enough attention over the past year to reach a successful conclusion, but I have made some progress.

The NEWT logic design is complete, and it fits in the target FPGA device (an Actel APA075 in a 144-pin TQFP package, for the technically inclined). Because the target FPGA operates with a 2.5V core voltage and a 3.3V I/O voltage, making a compatible system board requires an external (to the FPGA) voltage translation layer of circuitry. This circuitry has been designed, but has not yet been entered into an electronic schematic form. In addition, some of the power-supply circuitry needs finalization (mainly the supply for the LCD driver voltages). I expect to finish these tasks once I have completely verified the CPU design.

I finally got around to building a test-bench for the design for my logic simulator in July of this year. This test-bench models the parallel memories that will be on the final CPU board, as well as the voltage translation circuitry mentioned above, and the power supply. With all of this in place, the NEWT successfully fetched its first instruction on August 8th. Verification of the instruction set then began in earnest, and is nearing completion.

Most of the instructions worked properly (or at least to my interpretation of properly) the first time I tried them. The ones that didn't suffered from the usual logic designer mistakes (logic polarity problems or off-by-a-clock-cycle problems). The keyboard scanner seems to be scanning properly, but I still need to verify the encoding, roll-over and keyboard flag operation. This will be time consuming because emulating the external keyboard behavior in a simulation test-bench is difficult.

Once I started simulating, I realized that there were a couple of fundamental flaws in the system design. The first flaw comes about because I have to power down the FPGA to conserve power when the calculator is off. Powering down the FPGA (the entire CPU) means that I don't have any place to store information that is supposed to be kept alive as long as the batteries are inserted, like the register address. (Remember that the register address is held in the individual RAM chips in the original 41 design.) I'll have to store this information in the external RAM. I'm still working on this issue. Since I'm close to the limit on the logic available in the FPGA, I may have to cut some other feature (like the four of the five different turbo modes) to make room. I hope that everyone will be happy with only the 50X turbo mode.

The second flaw is similar to the first. The NEWT design uses a memory management unit (MMU) to translate from the logical Nut-CPU addresses to the actual physical addresses for the parallel memories, and the MMU information is held in a section of the external RAM memory. So at power-up, the contents of the MMU are not going to be valid. To account for this, I will have to add one bit of memory (a latch) in the voltage-translation circuitry to mark the MMU information as invalid until the operating system software is able to initialize the MMU. Once initialized, the problem goes away, because the RAM contents are kept alive by the battery. But the user may have to execute some kind of

"configure" program after the MEMORY LOST indication if I can't shoe-horn something into the 41 operating system.

Once the design was pretty-much working in normal 1X speed, I looked at the turbo mode. After a couple of false starts, the NEWT finally entered the 50X turbo mode on September 3rd. Once in turbo mode, however, there were a couple of problems. First, when the CPU jumped to a memory address that wasn't in the on-board memory it didn't actually slow down to 1X speed and fetch the instruction over the external bus. That was easily fixed, but then the memory address wasn't output on the ISA bus to fetch the instruction from external memory. When this circumstance happens, the CPU is actually stopped, and it waits for the external timing to finish the (fake) external transaction in progress. The latched address is then supposed to be output on the ISA bus and the CPU is restarted in synchronization with the external bus timing. The only problem is that I had the address latch clocked by the CPU clock, which isn't running because it's waiting to become synchronized with the external timing! Needless to say this is now fixed and the design seems to be entering an exiting the turbo mode like it's supposed to.

If anyone is interested, I've posted some simulation traces on-line, along with the NEWT technical manual, at www.systemyde.com/hp41 . The traces show several interesting cases of the NEWT actually operating in the simulator. I'll keep updating the site as the debug continues.

So what's left? Well, the end of the design of the hardware is on the horizon, assuming that I can continue to find spare time. Board fabrication and in-system testing comes next. Then I'll need to write some mcode to actually manage the memory and turbo mode operation. And I'll need to go through the operating system to identify any sections that might have timing dependencies so that they can be marked for 1X operation (tone, keyboard debounce, etc.).

At this point, I think that I've shown that it doesn't take millions of dollars to move a design to a new technology. After all, I've only spent a grand total of about 250 hours on this project, and the only real cost has been to print a couple of copies of the technical manual. What about the case where someone wanted to actually fabricate a custom chip? This design is less than 5000 logic gates. The NRE for a 0.35u gate array (why would you go any smaller, being pad-limited?) would run about $50k and the chips themselves would cost less than a dollar. Q.E.D.

Postscript:

I designed for 18MHz, or 50x the original Nut frequency because I didn't have a good feel for how fast the FPGA would run. It turns out that the design can actually run at 25MHz in the FPGA (70x), but I don't plan on going back and changing anything at this point. Based on a test synthesis run that I did for a 0.35u implementation, I expect that it would be possible to run the design at over 60MHz (170x) in the gate array I mentioned above. Imagine...