



Will Self-timed Asynchronous Logic Rescue CPU Design?

By Bernard Cole
August 24, 2002

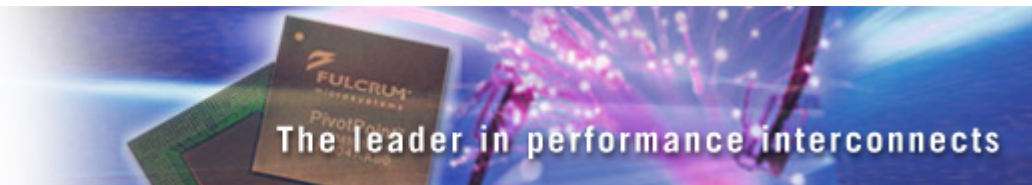
Like the proverbial reaction of a bull to a red flag, the mere mention of asynchronous logic design prods engineers into heated debate. It will be interesting to see how the computer industry reacts to the revelation this month at the Hot Chips Symposium at Stanford University by Fulcrum Microsystems, (Calabasas, CA) of a MIPS-based self-clocking asynchronous design. However much debate the announcement provokes, will the topic and the issues it raises quickly disappear, as has happened several times in the past? Or will it actually trigger some industry-wide soul-searching about new design methodologies? Based on the indifference, boredom, and even outright hostility to this idea in the past, my hunch is that the idea will just disappear. But I hope that this methodology, or something like it, finally moves into the mainstream of computer design. Something is needed to pull in the reins on the synchronous horse the industry is riding. Otherwise, we will all ride it right into a wall of inoperability caused by outrageously high power consumption, power dissipation, and global clocking hang-ups as ICs get bigger, denser, and faster.

Illustrative of the lack of public attention given the subject, the presentation by Fulcrum founder and chief technical officer Andrew Lines was the only one of 25 papers at the symposium that addressed the topic. I wonder if the fact that Fulcrum has built a crossbar switch at the heart of its asynchronous design capable of a 260-gigabit-per-second bandwidth in a standard, and relatively conservative, 0.18 micron (180 nanometer) CMOS process sparks any interest.

Unlike the familiar design methodologies used now, self-clocking, asynchronous circuits remove the need for a global synchronizing clock. Instead, the process of computation is controlled through local clocks and local handshaking and handoff between adjacent units. What this means for high performance and power-efficient design is that such local control permits resources to be used only when they are necessary, similar to data flow architecture in the highly parallel designs used in embedded network processors. Although asynchronous designs usually require more transitions on a computational path than synchronously designed CPUs, the transitions usually only occur in areas involved in the current computational task.

Intuitively, the methodology offers a lot of benefits to designers of the new generation of ubiquitously connected embedded devices and small footprint appliances as well as many embedded networking designs. Because self-clocking allows more fine grained control over the logic, only those portions of the chip that are actively involved in a particular operation are on at any one time. This means that the majority of resources of a circuit can be focused on that chore without driving up power dissipation and consumption, the limiting factors in almost any limited resource design.

With regards shifting to asynchronous design, the electronics and computer industry reminds me of the automobile industry as cars got bigger and faster, but also consumed more fuel and generated more pollution. Then the world started changing. Because of a constellation of factors, the public wanted smaller, more fuel efficient vehicles. While auto makers in the U.S. did finally get the message, they are still dragging their feet, preferring to come up with modifications and enhancements of the existing internal combustion engine.



Just as customers 30 years ago stopped buying big American cars, so have consumers stopped buying 2- and 3-GHz desktop systems. Computer and microprocessor vendors, like their auto making predecessors, are confronting a new world, and, like the husband who cannot understand why his wife has just left him, asks: "What did I do? What did I say?"

For auto makers, 30 years later, it is only now sinking in what they have to do: they are shifting away from gas-only engines, and committing to designs based on hydrogen, battery, power cell and hybrid technologies. Can we afford to wait another 30 years before the computer industry finally bites the bullet and tries something else -- anything else -- to power our smaller and faster, but more efficient, netcentric information engines?

Asynchronous circuit designs were common in the early days of circuit design and can still be found here and there, not as an all-encompassing design methodology, but as a problem solver, after every synchronous design trick has run out of gas.

By and large, most circuit designs are built with synchronous logic, small blocks of combinatorial logic separated by synchronously clocked registers. The biggest advantage, at SSI, MSI and LSI levels, at least, was that synchronous logic made it easy to determine the maximum operating frequency of a design by finding and calculating the longest delay path between registers in a circuit. AT VLSI and now at the ultra-scale levels with millions of gates it is becoming extraordinarily difficult, requiring much manpower, and computer time, to find and predict the critical path delays.

But that is not the only problem. For one thing there are the large clock current surges necessary, which tax a circuit's power distribution nets, as well as the thermal stability of the circuit. There is also the growing inability to control noise and metal integrity. And in some of today's system on chip designs with not only millions of gates, but millions of heterogeneous gate structures -- DRAM, SRAM, ROM, flash, digital logic, programmable logic and analog circuitry -- the job of maintaining the global clocking across the area of a chip is not easy.

To be sure, process engineers and logic designers have been quite innovative in finding quick fixes when their synchronous engines show signs of sputtering and dying. They remind me of when my cousin Jerry and I would race around the farm country in rural Oklahoma in his carefully built hot rod. Invariably it would at some point give out and die, whereupon Jerry would jump out, look at the engine and say, with just a little edge of panic in his voice: "It's OK. Don't worry. I got it. I got it." He would find a clever fix and we would be on our way again, but we were unlikely to get anywhere on schedule.

As we push process technology further down though the micrometer to the nanometer level, things only get worse, with shot noise, charge sharing, thermal effects, supply voltage noise and process variations all making calculations of delay that more uncertain and difficult. Although it is being resisted fiercely as an all-encompassing design methodology, major semiconductor firms such as Intel, TI, IBM, and LSI Logic are selectively making use of asynchronous self-clocking mechanisms in their designs.

Start ups and research efforts at universities continue to work on full designs and the supporting infrastructure to make asynchronous logic more acceptable to mainstream computer design. Several years ago, a research



team at the University of Manchester in the UK was able to build a fully asynchronous ARM processor with the performance equivalent of a traditionally designed CPU, and at much lower power.

Even those early designs made it clear, at least to me, that asynchronously designed circuits would be of enormous benefit in what is now emerging as the mainstream of computing in small footprint iappliance design, and many areas of embedded networking, where power issues and reliability are just as important, if not more so, than pure speed.

If the benefits are so clear, why does this "not so new" asynchronous methodology, which was covered extensively in Carver Mead's seminal Introduction to VLSI Systems (Addison-Wesley, 1980), engender so much emotion when I bring up the topic in conversations with engineers?

I don't think it is a technical issue, but an infrastructure support problem. It's the chicken-or-egg question all over again: we cannot easily design asynchronous systems because appropriate tools aren't available. And there are no tools, the EDA houses say, because there is no demand for them.

Break outs from this destructive closed loop are occurring, as basic research continues on tool methodologies at schools such as the University of Manchester, the University of Washington, California Institute of Technology and the University of Utah.

Fulcrum itself is trying to break out of the loop by coming up with a family of MIPS-based SOC cores, which while operating asynchronously internally, will allow developers to design the cores into their synchronously-designed SOCs.

What also makes me optimistic is that EDA tool companies seem to be abandoning their "better the evil you know rather than the evil you don't" attitude. Mentor Graphics over the last few years has nosed around the edges of the problem in a number of internal studies. And Cadence Design Systems is an investor in Fulcrum and a partner with the company in the development of the EDA tools Fulcrum is using.