DOI:10.1145/2347736.2347749

Ivan Sutherland

Viewpoint
The Tyranny of the Clock

Promoting a clock-free paradigm that fits everything learned about programming since Turing.

In Alan Turing's day logic was slow and costly but, relative to logic, wires were fast and almost free. Since then the costs of logic and wires have reversed: modern logic is fast and almost free but, relative to logic, wires are now slow and costly. They are costly in three ways. 1) The wires in a modern circuit chip cost most of its area; the transistors in a chip hide underneath a thick bed of tiny wires. 2) The wires in a modern circuit chip cost most of the delay. 3) Worst of all, moving electric charge onto and off of wires wastes most of the energy. The cost of logic and memory dominated Turing's thinking, but today, communication rather than logic should dominate our thinking.

Does communication dominate your thinking?

My question applies equally to hardware, software, and theory.

Today's digital design paradigm, the "clocked" design paradigm, depends on a rhythmic clock signal. The clock signal breaks time into discrete time steps. The designer knows exactly his intent for all the actions of each time step and can check that all the necessary precursors for the actions of each time step happen in earlier steps. Discrete time steps simplify the design task.

Before the telegraph, there was no easy way to synchronize time over distance. Fortunately, there was little need outside navigation to know what time it is somewhere else. "Simultaneous" did not need to apply between Chicago and New York; each city could be its own time zone. The railroad

changed that: passengers wanted to know at what local time their train would arrive, and dispatchers wanted to avoid collisions. Fortunately, the telegraph could provide a notion of "simultaneous" from New York to Chicago so that schedules could be kept. Like a railroad the clocked design paradigm makes designers *want* a concept of "simultaneous" so that clock periods can begin and end everywhere simultaneously.

As transistors and wires get smaller, the area over which one can deliver a clock signal "simultaneously" becomes smaller and thus the number of "clock zones" must increase. The clock beat of each zone differs from the beat of its neighbors in phase and often in frequency as well. A large chip may have hundreds or even thousands of separate clock zones. The clocked design paradigm helps *within* each zone, but only within the zone.

Between clock zones the clocked design paradigm retards data flow. The clocked design paradigm insists on synchronizing all incoming data to the frequency and phase of the clock in the destination zone. Synchronizing a data signal to the destination clock requires special precaution against errors. A reliable boundary crossing requires a delay of two or three clock periods. It is as if each clock zone posted customs inspectors at its borders. The clocked design paradigm exacerbates communication delay.

Ever since my 1988 Turing lecture, I have been exploring an alternative "clock-free" design paradigm. I seek change in the design paradigm to cast off the tyranny of the clock. Instead of making all logic "march to an external drum beat," let us allow each logic element to proceed at its own pace. Because each element acts only when and if necessary, such a paradigm shift will lead to designs that save energy. The clock-free paradigm will also make computers go faster because doing away with border-crossing delays speeds communication. I see a parallel to the economic efficiency Europe gains from free communication across national borders.

PHOTOGRAPH BY ERIK VELLDAL

Casting off the tyranny of the clock offers freedom to optimize the separate parts of a design. For example, Rajit Manohar and his students at Cornell report a clock-free IEEE-compliant, double precision, floating-point adder with the same throughput as an equivalent clocked design. The Cornell clock-free design uses less than half, about 40%, as much energy per addition as its clocked counterpart. The Cornell design gains simplicity and thus reduces energy by doing easy cases fast and allowing the rare hard cases to take longer. A recent paper from my group in the Asynchronous Research Center at Portland State University reports on faster division by allowing steps that merely shift to go faster than steps that must subtract.

Casting off the tyranny of the clock offers modularity as well as local optimization. Sam Fuller, then chief engineer at Digital Equipment Corporation, once told me that his process people could provide faster chips every six months. He complained that his product could not similarly improve every six months because it took 18 months to redesign an entire computer for the new clock speed. The tyranny of the clock made his design insufficiently modular to permit incremental improvement. *He chose* to march his entire machine to a single drumbeat rather than allowing each part to work at its own best speed.

Like all tyranny, the tyranny of the clock stems from the range over which *we choose* to subject *ourselves* to the tyrant's authority.

The clock-free paradigm I promote relates to the clocked design paradigm as a "free economy" relates to a "controlled economy." We can regain the efficiency of local decision making by revolting against the pervasive beat of an external clock.

Clock-free commercial products are in use today. Handshake Solutions, a computer-aided design company from the Netherlands, was proud of having 700 million of their clock-free chips in use in smart cards, passports, cellphones, and other portable devices. Fulcrum Microsystems, a Caltech spinoff recently purchased by Intel, sells a self-timed communication switch with outstanding performance.

Casting off the tyranny of the clock offers freedom to optimize the separate parts of a design.

The paradigm shift I seek faces three formidable obstacles: technical, social and courage. First, technical: Make no mistake; designing a clock-free system can face the same hard problems of parallelism that give software people nightmares. But a few pioneers have shown that clock-free design is possible and sometimes even easy. The pioneers have uncovered benefits like using less than half, 40%, of the energy per operation as reported by Cornell. Second, social: *All* of today's commercial design tools assume clocked design. All engineering schools teach clocked design. Will we ever train enough young people in the clock-free paradigm for it to self-perpetuate? Third, courage: Management knows the costs, difficulties, and results of the "tried and true" clocked design paradigm. Management chooses "to bear those ills we have rather than fly to others that we know not of."

The clock-free design paradigm must eventually prevail. It fits physics. Each increase in the relative cost of communication over logic brings us closer to the fundamental physical truth that "simultaneous" lacks meaning. The clock-free paradigm fits everything we have learned since Turing about programming. Software avoids tyrannous global time constraints. Without freedom from global time constraints, software libraries would be impossible. "Modularity" and "data hiding" are basic principles of quality software because they allow reuse and local optimization. Software is self-timed: Each subroutine runs at its own pace; its users wait for it to finish. Imagine what software would be like if subroutines could start and end only at preset time intervals. "My subroutines all start at 3.68 millisecond intervals; how often do yours start?"

Software development proceeds from correctness to performance. After software works, we tune its heavily used parts to achieve the desired performance. Performance almost always depends on only a small part of the whole. Compare this to the situation in a clocked hardware design where each and every signal must arrive "on time," even if it is rarely used. The tyranny of the clock wastes both engineering cost at design time and energy at runtime. What a needless waste!

Only a small handful of intrepid entrepreneurs and academic researchers have yet dared to explore the clock-free paradigm I promote. I predict that sometime soon, some courageous management will tire of wasting money on the tyranny of the clock and adopt the clock-free design paradigm. Such courage will reap giant rewards. I shall be disappointed but not at all surprised if that courageous management speaks an Asian language rather than English, the native tongue I share with Alan Turing.

This Viewpoint is derived from Ivan Sutherland's presentation at the ACM A.M. Turing Centenary Celebration Computer Architecture panel discussion this past June; see http://amturing.acm.org/acm_tcc_webcasts.cfm. [Also see the profile of Ivan Sutherland in the News section of this issue on page 10. —*Ed.*]

Further Reading

Sutherland, I.E.
Micropipelines. *Commun. ACM 32*, 6 (June 1989).

Sutherland, I.E. and Ebergen, J.
Computers without clocks. *Scientific American 287*, 2 (Aug. 2002), 62–69.

Riaz Sheikh and R. Manohar.
An operand-optimized asynchronous IEEE 754 double-precision floating-point adder. In *Proceedings of the IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)* 2010, 151–162.

N. Jamadagni and J. Ebergen.
An asynchronous divider implementation. In *Proceedings of the IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2012.

Ivan Sutherland (ivans@cecs.pdx.edu) is a visiting scientist in the Asynchronous Research Center at Portland State University in Oregon and the 1988 recipient of the ACM A.M. Turing Award for his pioneering contributions to computer graphics.