Peter J. Denning

The Choice Uncertainty Principle

It is impossible to make an unambiguous choice between near-simultaneous events under a deadline. How can computations choose reliably?

In the early 1970s, as he was building the University of Cambridge CAP computer, hardware engineer David Wheeler sought to solve an old conundrum: computer logic circuits occasionally locked up, requiring a complete restart of the computer. These lockups could cause the loss of considerable work on long computations. Worse, they made machines unreliable for real-time, safety-critical applications. Wheeler wanted his hardware to be loss-free and reliable.

The lockup that concerned Wheeler was not a software issue, such as the modern "blue screen of death" in Windows or the "spinning beach ball" in Mac OS. The hardware itself locked up at random every few days even when running fully verified software. Some hardware engineers called the problem "cosmic ray crashes" because it seemed that only a random intervention from the universe could cause such behavior.

Wheeler noticed the lockups never occurred when the interrupts were turned off. Interrupt signals were recorded on a flipflop the CPU consulted between instructions: the CPU decided either to enter the next instruction cycle or to jump to a dedicated subroutine that responded to the interrupt signal. Wheeler suspected the timing of the interrupt signal's arrival to that flipflop occasionally caused it to misbehave and hang the computer. Imagine that: the simplest, most fundamental memory circuit of a computer could malfunction.

The Half Signal

A digital machine consists of storage elements interconnected by logic circuits. The storage elements, implemented as arrays of flipflops, hold the machine's state. The machine operates in a cycle: (1) Flipflops enter a state; the switching time is 10^{-12} to 10^{-15} seconds. (2) The logic circuits take the state as input and produce a new state; the propagation time of all inputs through the circuits is slower, 10^{-9} to 10^{-10} seconds. (3) The new state is read into the flipflops. A clock sends pulses that tell the flipflops when to read in the next state.

HAL MAYFORTH

Unambiguous choices between near-simultaneous signals must be made in many parts of computing systems.

The clock cycle must be longer than the propagation delay of the logic circuits. If it is any shorter, the inputs to some flipflops may still be changing at the moment the clock pulse arrives. Think about this. If an input voltage is changing between the 0 and 1 values at the time the clock pulse samples it, the flipflop sees a "half signal"—an in-between voltage but not a clear 0 or 1. Its behavior becomes unpredictable. A common malfunction is that the flipflop ends up in the wrong state: the clock-sampled value of an input intended to switch the flipflop to 1 might not be strong enough, so the flipflop remains 0.

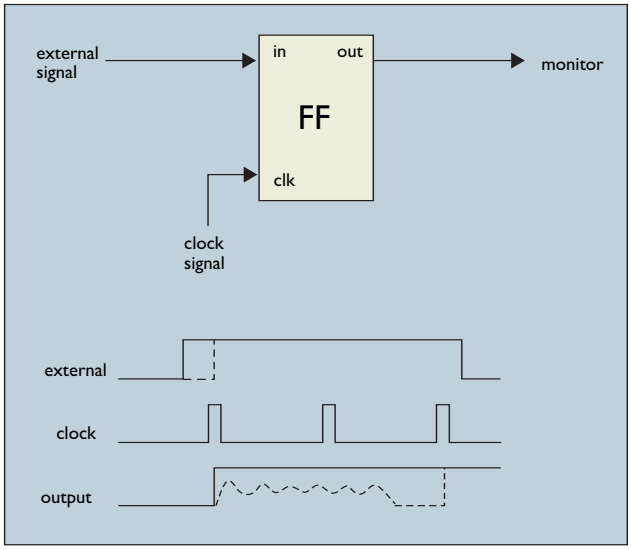

Figure 1. Experimental setup for observing flipflop (FF) metastability. Each clock pulse triggers the FF state to match the input signal. If the input signal is changing when the clock pulse arrives, FF may enter an indefinite state that lasts more than one clock interval (dashed lines). The test repeats cyclically after the external signal returns to 0. To maximize metastable events, the clock frequency is tuned to a multiple of the external signal frequency. In a digital computer, the indefinite output becomes the input of other logic circuits at the next clock pulse, causing half-signal malfunctions.

The Metastable State

Unfortunately, there is a worse malfunction. A half-signal input can cause the flipflop to enter a "metastable state" for an indeterminate time that may exceed the clock interval by a large amount. The flipflop eventually settles into a stable state, equally likely to be 0 or 1.

A flipflop's state is actually a voltage that moves continuously between the 0 and 1 values. The 0 and 1 states are stable because they are attractors: any small perturbation away from either is pulled back. A flipflop switches because the input adds enough energy to push the state voltage closer to the other attractor. However, a half-signal input can sometimes leave the state voltage poised precisely at the midpoint between the attractors. The state balances precariously there until some noise pushes it closer to one of the attractors. That midpoint is called the metastable state. The metastable state is like a ball poised perfectly on the peak of a roof: it can remain there for a long time until air molecules or roof vibrations change its balance, causing it to roll down one side or the other.

In 1973, Chaney and Molnor at Washington University in St. Louis measured the occurrence rate and holding times of metastable states [2] (see Figure 1). By synchronizing clock frequency with external signal frequency, they attempted to induce a metastable event on every external signal change. They saw frequent metastable events on their oscilloscope, some of which persisted for 5, 10, or even 20 clock intervals. Three years later, Kinniment and Woods documented metastable states and mean times until failure for a variety of circuits [6].

In 2002, Sutherland and Eber-

gen reported that contemporary flipflops switched in about 100 picoseconds (10^{-10} sec) and that a metastable state lasting 400 picoseconds or more occurred once every 10 hours of operation [9].

Xilinx.com reports that its modern flipflops have essentially zero chance of observing a metastable state when clock frequencies are 200MHz or less [1]. At these frequencies, the time between clock pulses (five nanoseconds) is longer than all metastable events. But in experiments with interrupt signals arriving 50 million times a second, a metastable state occurs about once a minute at clock frequency 300MHz and about once every two milliseconds at clock frequency 400MHz. In a computer system generating 500 interrupts per second, approximately 1/100,000 of the experimental rate, these extrapolate to one interrupt-caused metastable state about every two weeks at 300MHz and about every three minutes at 400MHz.

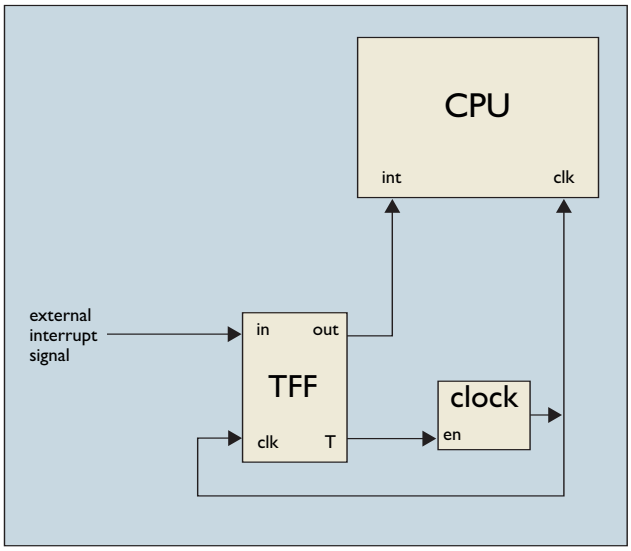

Figure 2. Threshold flipflop (TFF) output T is 1 when the state is 0 or 1 and is 0 when the state is metastable. Output T enables the clock. TFF can become metastable if the external interrupt signal changes just as the clock pulse arrives. Since the CPU does not run when the clock is off, it always sees a definite 0 or 1 when it samples for interrupts.

Wheeler's Threshold Flipflop

Wheeler, aware of the Chaney-Molnor experiments, realized that if the interrupt flipflop is driven metastable by an ill-timed interrupt signal, it can still be metastable at the next clock tick. Then the CPU reads neither a definite 0 nor 1 and can malfunction.

Wheeler saw he could prevent the malfunction if he could guarantee the CPU could read only the interrupt flipflop while it was stable. He made an analogy with the common situation of two people about to collide on a sidewalk. On sensing their imminent collision, both stop. They exchange eye signals, gestures, head bobs, sways, dances, and words until finally they reach an agreement that one person goes to the right and the other to the left. This could take a fraction of a second or minutes. They then resume walking and pass one another without a collision. The key is that the two parties stop for as long as is needed until they decide.

Wheeler designed a flipflop with an added threshold circuit that output 1 when the flipflop state was near 0 or 1. He used this for the interrupt flipflop with the threshold output wired to enable the clock to tick (see Figure 2). The CPU could not run as long as the interrupt flipflop was in a metastable state and thus could observe that flipflop only when it was stable.

With threshold interrupt flipflops, the University of Cambridge CAP computer achieved the reliability Wheeler wanted.

Arbiter Circuit

Unambiguous choices between near-simultaneous signals must be made in many parts of computing systems, not just at interrupt flipflops. Examples:

- Two CPUs request access to the same memory bank;
- Two transactions request a lock on the same record of a database;
- Two external events arrive at an object at the same time;
- Two computers try to broadcast on an Ethernet at the same time;
- Two packets arrive together at the network card;
- An autonomous agent receives two request signals at the same time; or
- A robot perceives two alternatives at the same time.

In each case, a chooser circuit

must select one of the alternatives for immediate action and defer the other for later action. There is no problem if the signals are separated enough that the chooser can tell which one came first. But if the two signals are near simultaneous, the chooser must make an arbitrary selection. This selection problem is also called the arbitration problem, and the circuits that accomplish it are called arbiter or synchronizer circuits [3, 8]. Ran Ginosar gives a nice account of modern synchronizers in [4].

state that can be induced in the chooser by conflicting forces generated when two distinct signals change at the same time.

In 1984, Leslie Lamport stated this principle in a slightly different way: "A discrete decision based upon an input having a continuous range of values cannot be made within a bounded length of time" [7]. He gave numerous examples of decision problems involving continuous inputs with inherent uncertainty about decision time. He argued that the

of the standard deviations of position and momentum is lower-bounded by a number on the order of 10^{-34} joule-seconds. Therefore an attempt to reduce the uncertainty of position toward zero may increase the uncertainty of momentum; we cannot know the exact position and speed of a particle at once. This principle manifests at quantum time scales and subatomic particle sizes—consider how small that bound is—but does not say much about macro effects of millions of elec-

If we try to force the choice before the process exits a metastable state, we are likely to get an ambiguous result or no choice at all.

The arbiter incorporates circuits, as in the Wheeler flipflop, that prevent it from sending signals while in a metastable state. Therefore all entities interacting with the arbiter will be blocked while the arbiter is metastable, and there is no need to stop a clock. The main effect of the metastable state is to add an unknown delay to the access time to the shared entity.

The Uncertainty Principle

We can summarize the preceding analysis as the choice uncertainty principle [5]: "No choice between near-simultaneous events can be made unambiguously within a preset deadline." The source of the uncertainty is the metastable

source of the uncertainty was the attempt to base a discrete decision on a continuous signal. We disagree on this point. Many continuous electronic circuits perform discrete decisions within bounded times; for example, a diode passes current in one direction but not the other. I have argued here that the source of uncertainty is the decision procedure itself. A device that selects among alternatives can become metastable if the signals denoting alternatives arrive at nearly the same time.

It might be asked whether there is a connection between the choice uncertainty principle and the Heisenberg Uncertainty Principle (HUP) of quantum physics. The HUP says that the product

trons flowing in logic circuits.

The HUP is sometimes confused with a simpler phenomenon, which might be called the observer principle. This principle states that if the process of observing a system either injects or withdraws energy from the system, the act of observation may influence the state of the system. There is therefore uncertainty about whether what is observed is the same as what is in the system when there is no observer. The observer principle plays an important role in quantum cryptography, where the act of reading the quantum state of a photon destroys the state. The information of the state is transferred to the observer and is no longer in the system.

The choice uncertainty principle is not an instance of the Heisenberg principle because it applies to macro-level choices as well as microscopic circuit choices. Neither is it an instance of the observer principle because the metastable state is a reaction of the observer (arbiter) to the system and does not exchange information with the system. (Neither is it related to the Axiom of Choice in mathematics, which concerns selecting one representative from each of an infinite number of sets.)

Choice Uncertainty as a Great Principle

The choice uncertainty principle is not about how a system reacts to an observer, but how an observer reacts to a system. It also applies to choices at time scales much slower than computer clocks. For example,

- A teenager must choose between two equally appealing prom invitations.
- Two people on a sidewalk must choose which way each goes to avoid a collision.
- A driver approaching an intersection must chose to brake or accelerate on seeing the traffic light change to yellow.
- The commander in the field must choose between two adjutants, both demanding quick decisions on complex tactical issues at different locations.
- A county social system must choose between a development plan that limits growth and one that promotes growth.

These examples all involve perceptions; the metastable (indecisive) state occurs in single or interacting brains as they try to choose between equally attractive perceptions. At these levels, a metastable (indecisive) state can persist for seconds, hours, days, months, or even years.

The possibility of indefinite indecision is often attributed to the 14th century philosopher Jean Buridan, who described the paradox of the hungry dog that, being placed midway between two equal portions of food, starved [3]. (Some authors use the example of an ass (donkey) instead of a dog; but it's the same problem [7, 9].) If he were discussing this today with cognitive scientists, Buridan might say that the brain can be immobilized in a metastable state when presented with equally attractive alternatives.

At these levels is it not normally possible to turn off clocks until the metastable state is resolved. What happens if the world is impatient and demands a choice from a metastable chooser? A common outcome is that no choice is made and the opportunities represented by the choices are lost. For example, the teenager gets no prom date, the pedestrians collide, the driver runs a red light, the commander loses both battles, or the county has no plan at all. Another outcome is that the deciding parties get flustered, adding to the delay of reaching a conclusion.

Conclusion

Modern software contains many external interactions with a network and must frequently choose between near-simultaneous signals. The process of choosing will always involve the possibility of a metastable state and therefore a long delay for the decision. Real-time control systems are particularly challenging because they constantly make choices under deadline conditions.

The metastable state can occur in any choice process where simultaneous alternatives are equally attractive. In that case, the choosing hardware, software, brain, or social process cannot make a definitive choice within any preset interval. If we try to force the choice before the process exits a metastable state, we are likely to get an ambiguous result or no choice at all.

The choice uncertainty principle applies at all levels, from circuits, to software, to brains, and to social systems. Every system of interactions needs to deal with it. It therefore qualifies as a Great Principle.

It is a mistake to think the choice uncertainty principle is limited to hardware. Suppose your software contains a critical section guarded by semaphores. Your proof that the locks choose only one process at a time to enter the critical section implicitly assumes that only one CPU at a time can gain access to the memory location holding the lock value. If that's not so, then occasionally your critical section will

fail no matter how careful your proofs. Every level of abstraction at which we prove freedom from synchronization errors always relies on a lower level at which arbitration is solved. But arbitration can never be solved absolutely.

Therefore, software's assumption that variables denoting alternatives are well defined and unchanging when we look at them is not always valid. The choice uncertainty principle warns us of this possibility and helps to manage it. C

References

1. Alfke, P. Metastable recovery in Virtex-II Pro FPGAs. Technical report xapp094 (Feb. 2005); available from the Xilinx.com Web site.
2. Chaney, T.J. and Molnor, C.E. Anomalous behavior of synchronizer and arbiter circuits. *IEEE Transactions on Computers 22* (Apr. 1973), 421–422.
3. Denning, P. The arbitration problem. *American Scientist 73* (Nov.–Dec. 1985), 516–518.
4. Ginosar, R. Fourteen ways to fool your synchronizer. In *Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems*, IEEE (Feb. 2003); www.ee. technion.ac.il/~ran/papers/Sync_Errors_Feb03.pdf.
5. Great Principles Web site; cs.gmu.edu/cne/pjd/GP.
6. Kinniment, D.J. and Woods, J.V. Synchronization and arbitration circuits in digital systems. *IEE Proceedings 123*, 10 (Oct. 1976), 961–966.
7. Lamport, L. Buridan's Principle. Technical report (1984); research.microsoft.com/users/lamport/pubs/buridan.pdf.
8. Seitz, C.L. System timing, chapter 7 in *Introduction to VLSI Systems* (C. Mead and L. Conway, Eds.), Addison-Wesley, 1980, 218–262.
9. Sutherland, I. and Ebergen, J. Computers without clocks. *Scientific American* (Aug. 2002), 62–69; research.sun.com/async/Publications/KPDisclosed/SciAm/SciAm.pdf.

Peter J. Denning (pjd@nps.edu) is the director of the Cebrowski Institute for Information Innovation and Superiority at the Naval Postgraduate School in Monterey, CA, and a past president of ACM.

© 2007 ACM 0001-0782/07/1100 $5.00

COMING NEXT MONTH IN COMMUNICATIONS

Cover Story: Creativity Support Tools: Accelerating Discovery and Innovation

Creativity support tools work to join technology, science, and the arts to find common themes for user interaction and new media design. This article will examine how these tools accelerate discovery and innovation by cultivating and sustaining creativity. Moreover, such socio-technical environments empower the creative process and encourage creative mind-sets.

11th Century Common Law Meets the Internet
Are Your Citations Clean?
Advocating Research on the Datapath for Computing Disciplines
Self-Organization in Manufacturing Operations
Achieving Successful IS Design without User Participation
Strategies for Standards Development and Diffusion
Evolving Knowledge Management Problems
Tracing Variations in Software Product Families
Business Relationships through Better Decisions
The Role of Shared IT-Business Understanding