

## THE CHOICE UNCERTAINTY PRINCIPLE<sup>†</sup>

Peter J. Denning, Naval Postgraduate School, Monterey, California

January 2008

Rev 6/5/08

**Abstract:** The choice uncertainty principle says that it is impossible to make an unambiguous choice between near-simultaneous events under a deadline. This principle affects the design of logic circuits in computer hardware, real-time systems, and decision systems.

**Keywords:** uncertainty principles, half-signals, metastable state, arbiter, threshold flip-flop, choices under deadlines

The choice uncertainty principle says that it is impossible to make an unambiguous choice between near-simultaneous events under a deadline. This principle affects the design of logic circuits in computer hardware, real-time systems, and decision systems.

One of the first persons to notice that a fundamental principle might be at work in circuits that make decisions was David Wheeler of the University of Cambridge. In the early 1970s, he sought to build a computer whose hardware did not suffer from “hardware freezes” that were common in earlier computers. Wheeler noticed the lockups never occurred when the interrupts were turned off. Interrupt signals were recorded on a flip-flop 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. He suspected that the timing of the interrupt signal’s arrival to that flip-flop 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 flip-flops, hold the machine’s state. The machine operates in a cycle: (1) Flip-flops 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 flip-flops. A clock sends pulses that tell the flip-flops when to read in the next state.

---

<sup>†</sup>This article is adapted from the author’s article of the same title, *ACM Communications* 50:11 (November), 2007.

The clock cycle must be longer than the propagation delay of the logic circuits. If it is any shorter, the inputs to some flip-flops may still be changing at the moment the clock pulse arrives. If an input voltage is changing between the 0 and 1 values at the time the clock pulse samples it, the flip-flop 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 flip-flop ends up in the wrong state: the clock-sampled value of an input intended to switch the flip-flop to 1 might not be strong enough, so the flip-flop remains 0.

### **The Metastable State**

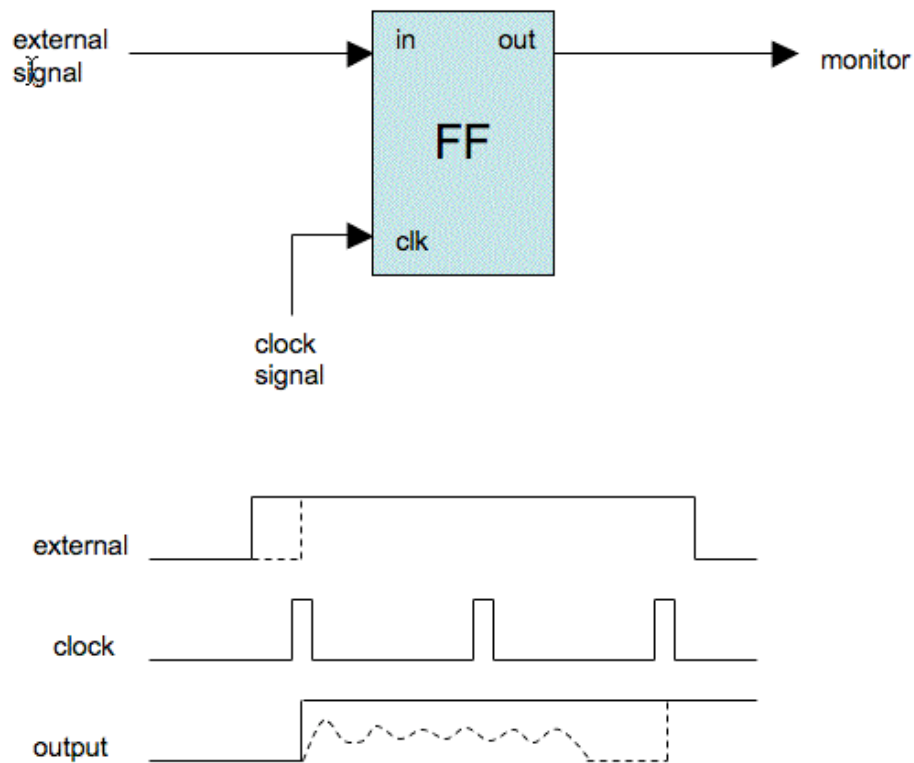
Unfortunately, there is a worse malfunction. A half signal input can cause the flip-flop to enter a “metastable state” for an indeterminate time that may exceed the clock interval by a large amount. The flip-flop eventually settles into a stable state, equally likely to be 0 or 1.

A flip-flop’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 flip-flop 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 sit there for a long time until air molecules or roof vibrations cause it to lose its balance, causing it to roll down one side of the roof.

In 1973, Chaney and Molnor at Washington University in St Louis measured the occurrence rate and holding times of metastable states (1); see Fig. 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 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 (2).

In 2002, Sutherland and Ebergen reported that contemporary flip-flops switched in about 100 picoseconds ( $10^{-10}$  seconds) and that a metastable state lasting 400 picoseconds or more occurred once every 10 hours of operation (3).

Xilinx.com reports that its modern flip-flops have essentially zero chance of observing a metastable state when clock frequencies are 200 MHz or less (4). At these frequencies, the time between clock pulses (5 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 a clock frequency of 300 MHz, and about once every two milliseconds at a clock frequency of 400 MHz.



**Fig. 1.** Experimental setup for observing flip-flop (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 (dotted 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.

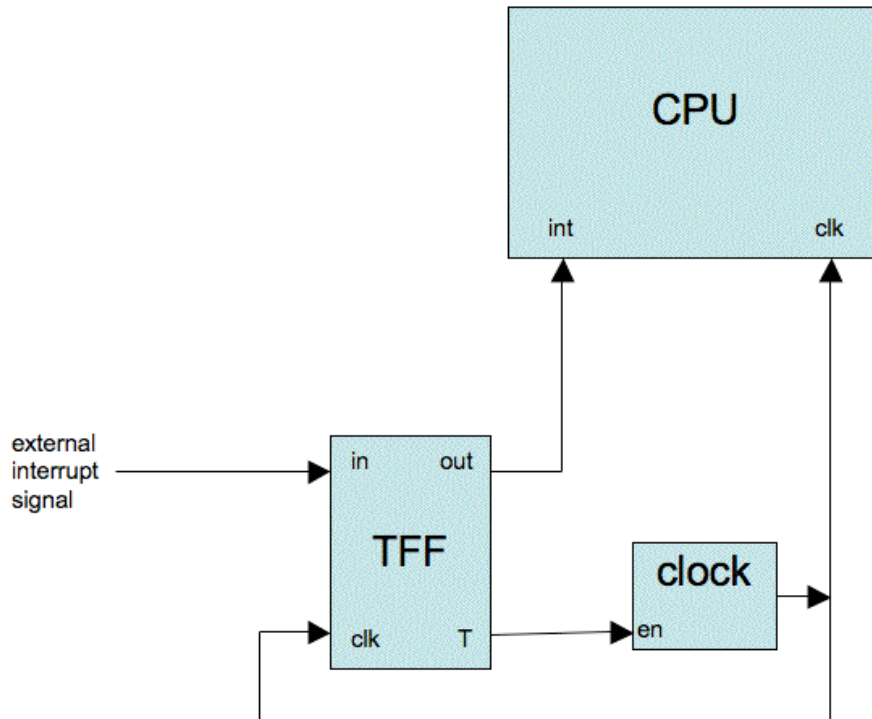
### Wheeler's Threshold Flip-flop

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

Wheeler saw he could prevent the malfunction if he could guarantee that the CPU could only read the interrupt flip-flop 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, they 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 or 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 flip-flop with an added threshold circuit that output 1 when the flip-flop state was near 0 or 1. He used this for the interrupt flip-flop with the threshold output wired to enable the clock to tick (see Fig. 2). The CPU could not run as long as the interrupt flip-flop was in a metastable state, and thus it could observe that flip-flop only when it was stable.

With threshold interrupt flip-flops, Wheeler's computer achieved the reliability that he wanted.



**Fig 2.** Threshold flip-flop (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.

### Arbiter Circuit

Unambiguous choices between near-simultaneous signals must be made in many parts of computing systems, not just at interrupt flip-flops. 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 to the network card.

- An autonomous agent receives two request signals at the same time.
- 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 (5,6). Ran Ginosar gives a nice account of modern synchronizers (7).

The arbiter incorporates circuits, as in the Wheeler flip-flop, that will 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 analysis above as the Choice Uncertainty Principle (8): No choice between near-simultaneous events can be made unambiguously within a preset deadline. The source of the uncertainty is the metastable 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 (9). He gave numerous examples of decision problems involving continuous inputs with inherent uncertainty about decision time. The source of the uncertainty, however, is not necessarily the attempt to sample a continuous signal; it 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 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 -- look at how small that bound is -- but does not say much about the macro effects of millions of electrons 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 macrolevel choices as well as to 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 different, 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 choose 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 fourteenth century philosopher Jean Buridan, who described the paradox of the hungry dog that, being placed midway between two equal portions of food, starved (5). [Some authors use the example of an ass (donkey) instead of a dog, but it is the same problem (3,9).] 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 it is 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 deadlines.

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 qualifies, therefore, as a Great Principle.

It is a mistake to think that the choice uncertainty principle is limited to hardware. Suppose that 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 is 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.

## BIBLIOGRAPHY

1. T. J. Chaney and C. E. Molnor. Anomalous behavior of synchronizer and arbiter circuits. *IEEE, Transactions Comput.*, **22**: 421-422, 1973.
2. D. J. Kinniment and J. V. Woods, Synchronization and arbitration circuits in digital systems. *IEEE Proc.*, 961-966, 1976.
3. I. Sutherland and J. Ebergen, Computers without clocks, *Scientif. Am.*, 62-69, Aug. 2002. Available from Sun Microsystems, <http://research.sun.com/async/Publications/KPDisclosed/SciAm/SciAm.pdf> .
4. P. Alfke, Metastable recovery in Virtex-II Pro FPGAs. Technical Report xapp094 (Feb. 2005). Available from Xilinx.com website.
5. P. Denning, The arbitration problem, *Am. Scient.* **73**: 516-518, 1985. It is interesting that some authors ascribe the indecision paradox to an "ass", although Buridan's original text refers to a "dog".

6. C. L. Seitz, System timing, in C. Mead and L. Conway (ed.), *Introduction to VLSI Systems*. Reading, MA: Addison-Wesley, 1980, pp. 218-262.
7. R. Ginosar, Fourteen ways to fool your synchronizer". *Proc. 9th Int'l Symp. on Asynchronous Circuits and Systems*, IEEE, 2003, 8pp. Available: [http://www.ee.technion.ac.il/~ran/papers/Sync\\_Errors\\_Feb03.pdf](http://www.ee.technion.ac.il/~ran/papers/Sync_Errors_Feb03.pdf) .
8. Great Principles Web site: <http://cs.gmu.edu/cne/pjd/GP> .
9. L. Lamport, Buridan's principle. Technical report, 1984. Available: <http://research.microsoft.com/users/lamport/pubs/buridan.pdf> .