The Embedded Muse 42

Editor: Jack Ganssle (jack@ganssle.com) January 7, 2000

Metastability – Take 2

Many of you kindly wrote in with your comments about my metastability question: “can a metastable flip flop settle in a random state?” My email inbox has been swamped with your replies!

Special thanks to Larry Marks for faxing a number of articles about the subject.

There are two general themes in the articles and your replies:

Yes, the output of a flop operating in the metastable region is random. It will take perhaps a long time to settle into a state; that state will have no relation to the data at the flop’s D input.

The question I asked might not have a lot of meaning, since who knows what the correct state is? If the flop’s input changes just a nanosecond or so before a clock transition, philosophically the signal is transitioning anyway.

Eldridge Mount wrote:
In "Digital Design: Principles and Practices", second edition by John F. Wakerly, a rather helpful analogy is given to describe metastability on pages 451 & 452:

"Metastable behavior of a bistable element can be compared to the behavior of a ball dropped onto a hill. If we drop the ball from overhead, it will probably immediately roll down to one side of the hill or the other. But if it lands right at the top, it may precariously sit there for a while before random forces (wind, rodents, earthquakes) start it rolling down the hill..."

...and here's the kicker, a direct answer to your question:

"...Like the ball at the top of the hill, the bistable may stay in the metastable state for an unpredictable length of time before non-deterministically settling into one stable state or the other."

Jon Plotky mentioned a design he had worked on:

After the board had been shipping for a while, all of a sudden we started having failures with a certain batch of cards. We found that the problem cards all had Signetics 74S74 devices, while the working cards used other manufacturers. Even more interesting, the problem with the Signetics parts was not that it's metastable state lasted longer. The problem turned out to be that when the input setup time was violated it would sometimes cause improper operation of the OTHER flip-flop in the package! The output of the second flop would change when the first flop went metastable. I never received an explanation of this from Signetics.

Jerold Green gave his 5 rules of async design:

1. Asynchronous signals should be clocked with an additional flip flop running at the same frequency as the logic that it drives. The direct input should not be used at all. There should only be one such flip flop, since multiple flip flops monitoring the same signal may clock different data.
2. Two flip flops in series should be used if it is absolutely necessary to use this signal to drive asynchronous logic, such as an asynchronous reset or clock.
3. If you are 'pushing' the device speed, review the metastability data on the device regarding maximum metastable delay vs. the clock rate.
4. Know what signals are asynchronous to your system. If you generate multiple clocks to drive different functions, the interfaces between them can easily be asynchronous.
5. Verify setup delays. Synchronous system speeds are limited to the maximum logic delay plus setup time. A PLD may specify a high speed toggle rate for a shift register (no logic), but multi-level logic will limit the attainable speed.

Bas van Rossem summed it all up: “The bottom line is; never trust asynchronously received data.”

Many thanks to the many, many others who wrote in. Clearly this is a subject of concern to many of us!

So here’s a scenario that’s been bothering me lately. Suppose you’re latching parallel data that comes asynchronously. Perhaps a 12 bit word from an encoder. If you latch this data before reading it (to eliminate the problem of having the data change during the read cycle), then you create a possible metastable condition. That is, the latching signal may come just as data starts to change.

Now we’ve got up to 12 flip flops entering a metastable state. Since each output may be random, the CPU may see data that literally has no meaning. If it acts on this meaningless data, bad things are sure to result.

So I propose adding a sixth rule to Jerold Green’s list:

When a program reads asynchronous data, always read twice and compare the results to correct for potential metastable states.

Thought for the Week

Words for the New Millennium:

Blamestorming: Sitting around in a group discussing why a deadline was missed or a project failed, and who was responsible.

Chainsaw Consultant: An outside expert brought in to reduce the employee headcount, leaving the top brass with clean hands.

Cube Farm: An office filled with cubicles.

Ego Surfing: Scanning the Net, databases, print media and so on, looking for references to one's own name.

404: Someone who's clueless. "Don't bother asking him; he's 404." From the WWW error message "404 Not Found," meaning the requested document couldn't be located.

Idea Hamsters: People who always seem to have their idea generators running.

Keyboard Plaque: The disgusting buildup of dirt and crud found on computer keyboards.

About The Embedded Muse

The Embedded Muse is an occasional newsletter sent via email by Jack Ganssle. Send complaints, comments, and contributions to him at jack@ganssle.com.

To subscribe, send a message to majordomo@ganssle.com, with the words “subscribe embedded *your-email-address*” in the body. To unsubscribe, change the message to “unsubscribe embedded *your-email-address*”.