techXclusives

Get Smart About Reset (Think Local, Not Global)

By Ken Chapman
Staff Engineer, Core Applications - Xilinx UK

If you are the type of person that believes that the reason we don't all fall over is because the the world is flat, and that digital logic is only a case of "1's", "0's", and Boolean algebra, then you will be a lot happier if you do not read this TechXclusive. If reliable digital designs are important to you as devices get bigger and faster, then brace yourself and read on...

One of the commandments of digital design states:

"Thou shalt have a master reset for all flip-flops so that the test engineer will love you, and your simulations will not remain undefined for time eternal."

So, it may come as a shock to learn that applying a global reset to your FPGA designs is not actually a very good idea and is something you should generally avoid. Clearly, this is a very controversial issue, so let's take a look at the reasons why such a design policy should be considered.

But a global reset isn't timing-critical!

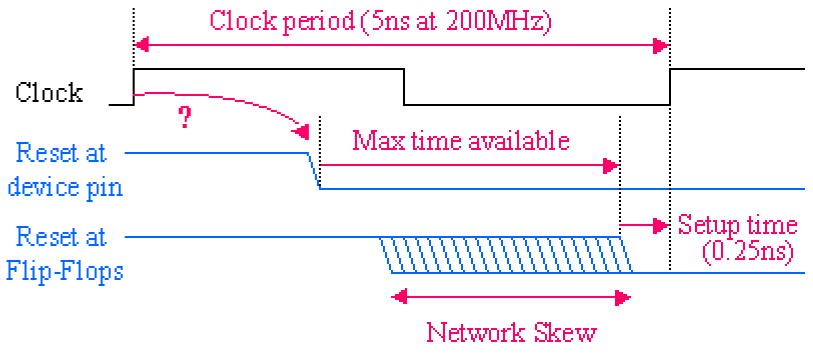
What are the typical drivers of a global rest signal?

- **Press switch**: Definitely slow and very undefined timing.
- **Power supply status output**: Active for a long period until supply stable.
- **Microprocessor**: Pulse tends to be long.

In all these cases, it would appear that the reset signal is slow; hence, it would also appear safe to assume that it is not timing critical. Indeed, when specifying a timing constraint for your Virtex™ design, this signal would normally be assigned a long period (low frequency).

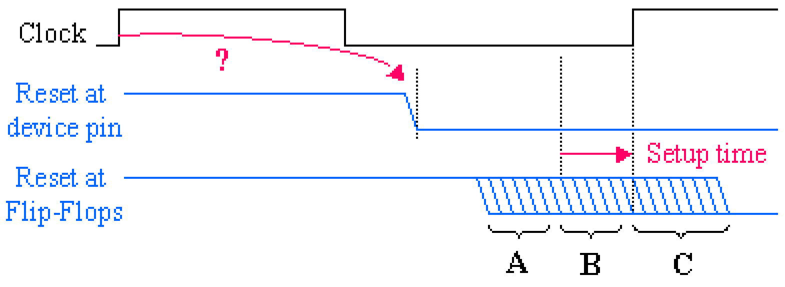
However, this is not strictly true, and statistically becomes a bigger issue as clock rates increase. Although the duration of the reset pulse may be long relative to the clock period and guarantee that all the device flip-flops are reset, the release of the reset signal should be considered to be a-timing critical event.

It is important to remember that the release of the Global Set/Reset (GSR) within the device is also a global reset, and just because it is part of the silicon device, it is not infallible. This is also a high fan-out network within the device, and whilst the start-up sequence can be synchronised to a "user clock", it cannot be synchronised to all clocks that you may have in one design. (Devices have multiple DLL/DCM modules, and each is capable of generating multiple clocks and clock phases).



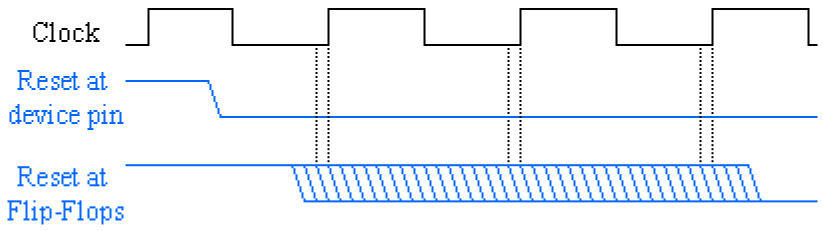
In the timing diagram above, we can see that a reset signal is de-asserted at some time between clock edges. The signal then propagates to the various flip-flops. At each flip-flop, the signal should be de-asserted a "set-up period" before the active clock edge. It is obvious that the higher the clock rate, then the less time available to distribute the reset signal. When you consider that the reset signal is a very high fan-out network, you realise what a huge challenge this is.

If the reset is released asynchronously to the clock (often the case), then there is no way to guarantee that all flip-flops will be released on the same clock, edge even if the distribution time is less than a clock period.



Flip-flops receiving the release of reset at "A" will be active on the first clock edge, but flip-flops receiving the release of reset at "C" will not become active until the following clock edge. The flip-flops at "B" are really hard to define and may even lead to metastability.

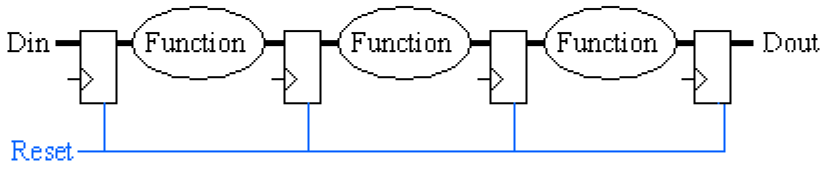
With increasing clock rates, and the potential distribution skew associated with large devices, then it becomes almost inevitable that not all flip-flops are released in preparation for the same clock edge.



But does it really matter?

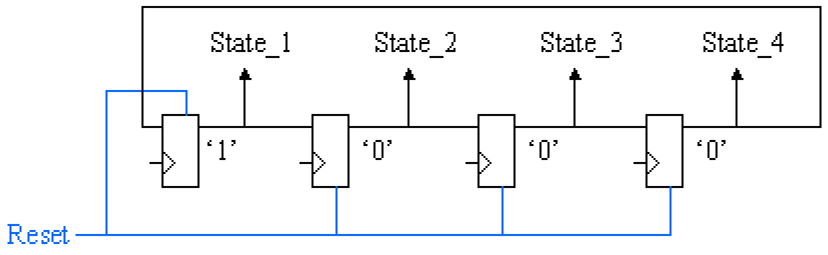
Well the good news is that in 99.99% of the time, it really doesn't matter. With statistics like that, it isn't surprising that most circuits work. But have you ever had one of those circuits that doesn't work the first time on a Tuesday afternoon?! If you have, then may be you have encountered one of the 0.01% cases and been unlucky enough to have released the reset at the wrong time.

Well, it doesn't matter here...



When there is a data flow through a pipelined process, it really doesn't matter when the master reset is released. After a few cycles, the entire pipeline will be operational, and any incorrect data will be flushed out of the system. In fact, there is little point in having a reset at all. Even a simulation will emulate the initial state following configuration, or unknown states will be purged out of the system as valid data inputs are applied.

But it does matter here...

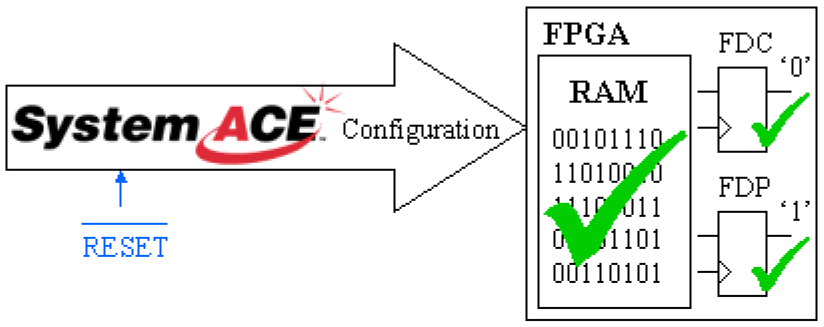


In this example of a simple "one-hot" state machine, there is a clear potential for failure. If the first flip-flop containing the "hot" state is released one clock cycle before the second flip-flop, then the "hot" state will be lost and the state machine will become 'cold' forever. The probability of this happening tends to be reduced by the close proximity of the flip-flops involved (low skew on localised reset network), but unless set-up time is guaranteed then it could still happen. It is also possible that an encoded state machine may transition into an unexpected state, including an illegal state, if all flip-flops are not released on the same clock cycle.

Ultimately, it is the circuits that contain a feedback path that require careful reset considerations. Circuits without feedback really do not need a reset at all. In digital signal processing applications, a finite impulse response filter (FIR) has no feedback. Output samples really have no value until valid data has filled all the taps, so resetting the tap registers achieves nothing. However, an infinite impulse response filter (IIR) contains feedback. If a spurious output sample is generated as a result of an unclean release of reset, then this will continue to effect output samples for a significant period of time. In the worse case, complete failure of the filter may occur due to instability.

Automatic coverage of the 99.99% of cases

When a Xilinx FPGA is configured, or reconfigured, every cell is initialised. This really is the ultimate in master reset because it covers far more than simple flip-flops.



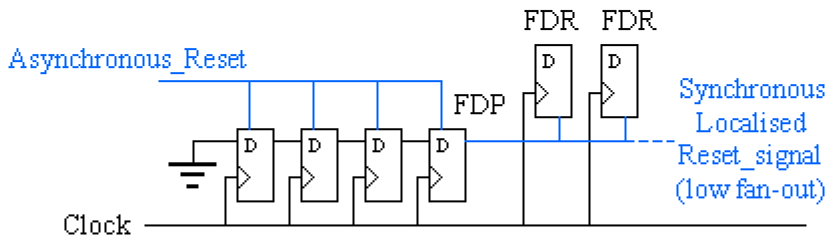
Configuration has the same effect as a global reset, but also initialises all RAM cells. With the increase in embedded RAM available on devices this is a very useful feature. To know that all RAM contents are pre-defined is ideal for simulation and operation and obviates the requirement to have "boot-up" sequences to clear memory.

As processors also become embedded in Xilinx devices (either hard or soft), all program and data areas are defined even before the processor executes the first instruction. With this natural benefit of Xilinx devices, it really doesn't make sense to burn up valuable programmable resources to reset only the flip-flops. Your simulator should be able to model this initialisation (often called a 'power-on reset') which again avoids any requirement for the reset signal in your design.

Strategy for the 0.01% of cases

The most important thing is that you should have a strategy for handling resets in your designs and cover these at design reviews. Establish which are the critical parts of the design that have to be released synchronously with an associated clock domain. A localised high performance reset network can then be inserted to control only those flip-flops that require it.

The following circuit shows a potentially useful mechanism to control a localised reset network. The circuit has the advantage of providing the same effect following device configuration as it does when an external signal is applied.



During configuration of the device, or an asynchronous reset signal, all flip-flops in the chain are preset to "1". Almost immediately, the last flip-flop of the chain then drives an active reset to the localised reset network. Following release of GSR or the asynchronous reset signal, the shift register chain begins to fill with "0" each clock cycle.

The number of flip-flops in the chain determines the minimum duration of the reset pulse issued to the localised network. Eventually, the last flip-flop makes the transition from high to low and the localised reset is released synchronously with the clock. Flip-flops being reset can employ synchronous set (FDS) or synchronous reset (FDR) leading to a fully synchronous design and easy timing specifications and analysis.

Reset costs more than you think!

The idea that having a global reset in your HDL code actually costs anything is generally far from the mind whilst implementing a design. However, the cost can actually be significant...

a) Routing resources of the device are used.
- Reduces freedom for other connections.
- May lower system performance potentially requiring a higher speed grade of device.
- Increased routing time (PAR).

b) Logic resources of the device are used.

- Use of dedicated reset on flip-flops.
- Operational resets result in additional gate before D input or dedicated reset input.
- May impact size of design.
- Additional logic level may impact system performance.
- Increased place and route time.

c) Prevents use of highly efficient features
- The SRL16E implements up to 16 flip-flops in each LUT.
- These flip-flops do not support reset and can not be used by synthesis tools when HDL specifies reset.
- Up to 16 times increase in size and product costs.
- Additional size potentially reduces system performance.
- Increased place and route times.

Summary

A design implemented in a Xilinx FPGA does not require you to insert a global reset network. For the vast majority of any design, the initialisation state of all flip-flops and RAM following configuration is more comprehensive than any logical reset will ever be. There is no requirement to insert a reset for simulation because nothing will be undefined. Since a Xilinx FPGA is already fully tested silicon, you won't be inserting scan logic in your design and running test vectors, so you will not need a global reset as part of this process either.

Inserting a global reset will impact your development time and final product costs even if these factors can not easily quantified. With the trend towards higher speed clocks and complete systems on a chip, the reliability issues must be taken seriously. The critical parts of a system that must truly be reset should be identified and the release of those resets on start up, or during operation, must be controlled as carefully as any other signal within a synchronous circuit.

As you are creating each section of your next design, simply ask yourself: "Does this bit need to be reset"?