WP272 (v1.0.1) March 7, 2008

Get Smart About Reset: Think Local, Not Global

By: Ken Chapman

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, some may be surprised to learn that applying a global reset to your FPGA designs is not a very good idea and should be avoided. Clearly, this is a controversial issue, so let's take a look at the reasons why such a design policy should be considered.

Global Reset Isn't Timing-Critical

What are the typical drivers of a global reset 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. When specifying a timing constraint for your FPGA design, this signal would normally be assigned a long period (low frequency).

However, the assumption that the global reset is not timing-critical is not strictly true, and this assumption 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.

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. While the start-up sequence can be synchronized to a "user clock," it cannot be synchronized to all clocks in one design. Devices have multiple DLL/DCM/PLL modules, and each is capable of generating multiple clocks and clock phases.

In Figure 1, 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 as the clock rate goes up, the time available to distribute the reset signal goes down. Considering the reset signal is a very high fan-out network, meeting the de-assertion timing requirements is a huge challenge.

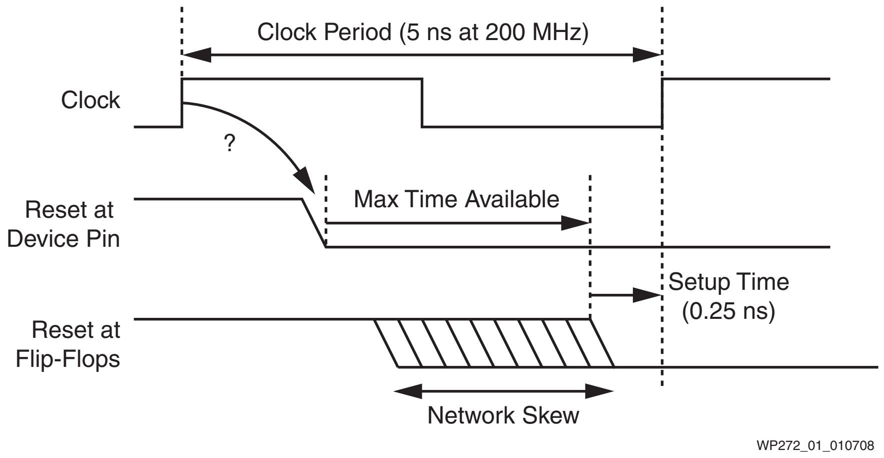

Figure 1: **Reset Timing Diagram - Asserted Between Clock Edges**

If the reset is released asynchronously to the clock (often the case), 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 (Figure 2).

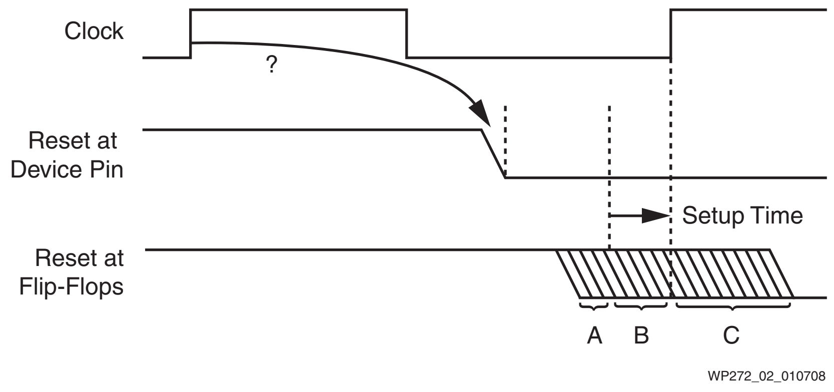

Figure 2: **Reset Timing Diagram - Asserted Asynchronously to the Clock**

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 difficult to define and may even lead to metastability.

With increasing clock rates and the potential distribution skew associated with large devices, it becomes almost inevitable that not all flip-flops are released in preparation for the same clock edge (Figure 3).

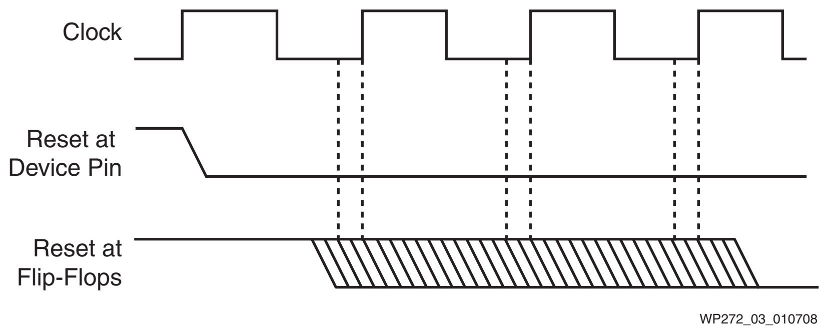

Figure 3: **Reset Timing Diagram - High Clock Rate**

Does It Really Matter?

The good news is that 99.99% of the time, the timing of the reset release really doesn't matter. With statistics like that, it isn't surprising that most circuits work. However, if you have ever had one of the circuits that doesn't work the first time, then maybe you have encountered one of the 0.01% cases and have been unlucky enough to have released the reset at the wrong time.

The timing of the reset release doesn’t matter in the scenario shown in Figure 4.

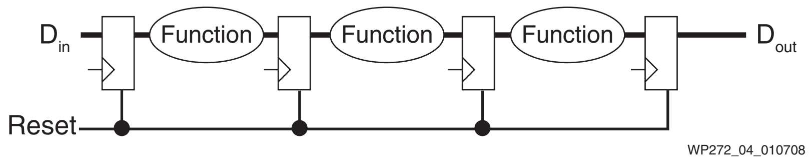

Figure 4: **Reset for a Pipeline**

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.

However, Figure 5 shows a scenario in which the timing of the reset release does matter.

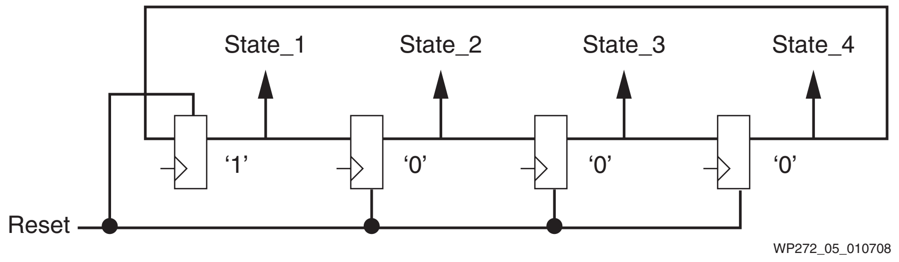

Figure 5: **Reset for a One-Hot State Machine**

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 localized reset network). However unless the set-up time is guaranteed, 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 feedback paths 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 the output samples will be affected for a significant period of time. In the worst 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 initialized (Figure 6). This is the ultimate in master reset because it covers far more than simple flip-flops.

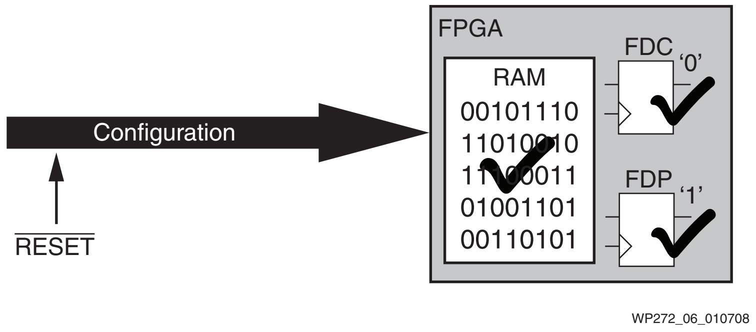

Figure 6: **FPGA Configuration**

Configuration has the same effect as a global reset, but it also initializes 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 eliminates 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 doesn't make sense to burn up valuable programmable resources to reset only the flip-flops. The simulator should be able to model this initialization (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 there is a strategy for handling resets in your designs, and that these strategies are covered at design reviews. Establish the critical parts of the design that have to be released synchronously with an associated clock domain. A localized high-performance reset network can then be inserted to control only those flip-flops that require a localized reset.

The circuit in Figure 7 shows a potentially useful mechanism to control a localized reset network. The circuit has the advantage of providing the same effect following device configuration as it does when an external signal is applied.

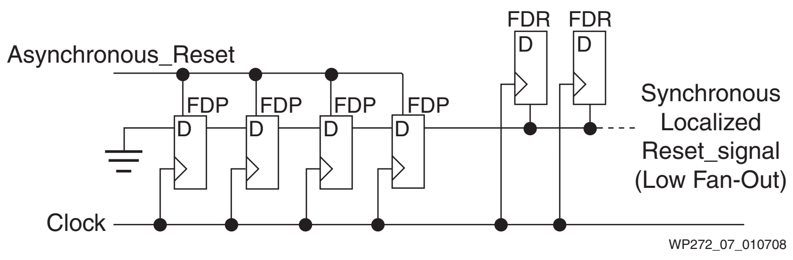

Figure 7: **Localized Reset**

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 localized reset network. Following release of GSR or the asynchronous reset signal, the shift register chain begins to fill with 0s each clock cycle.

The number of flip-flops in the chain determines the minimum duration of the reset pulse issued to the localized network. Eventually, the last flip-flop makes the transition from High to Low, and the localized reset is released synchronously with the clock. Flip-flops being reset can employ a 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!

While implementing a design, the costs of a global reset in HDL code can be overlooked. However, the cost can be significant:

- Routing resources of the device are used.
    - Reduces freedom for other connections.
    - May lower system performance potentially requiring a higher device speed grade.
    - Increased routing time.
- 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.
    - Almost certainly will impact size of design.
    - Additional logic level almost certainly will impact system performance.
    - Increased place and route time.
- Prevents use of highly efficient features, such as SRL16E.
    - The SRL16E implements up to 16 flip-flops in each LUT.
    - These virtual flip-flops do not support reset, and this prevents synthesis tools from delivering the full advantage offered by this feature when the HDL specifies a reset.
    - Up to 16 times increase in size and product costs.
    - Additional size potentially reduces system performance.
    - Increased place and route times.

For a discussion on how the priority of a reset signal can affect the resource utilization, see WP275 *Get your Priorities Right - Make your Design Up to 50% Smaller.*

Summary

A design implemented in a Xilinx FPGA does not require insertion of a global reset network. For the vast majority of any design, the initialization 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, scan logic and running test vectors are not necessary in the design. So, a global reset is not needed as part of this process either.

Inserting a global reset will impact development time and final product costs even if time and cost cannot be 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.

When creating each section of a design, simply ask, “Does this bit need to be reset”?

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
01/08/08	1.0	Initial Xilinx release. Some content taken from previous web postings as a TechXclusive.
03/07/08	1.0.1	Added author’s name

Notice of Disclaimer