What to do about Resets?

Andrew Wilson
andrew.e.wilson@byu.edu

***Abstract*—This paper presents advantages and disadvantages with resets in digital designs in both ASICs and FPGAs. When dealing with FPGAs, there are more details to consider when developing the reset network. Xilinx FPGAs have native power-on resets, flip flop primitives that support both types of resets and target active high resets, and a limited global reset network. FPGAs do not gain from the advantages of asynchronous resets as ASICs do. FPGAs should rely on the power-on reset to establish a known state throughout the device and use local synchronous resets when needed.**

***Index Terms*—resets, FPGAs**

I. Introduction

This paper will review the information released in these selected publication [1], [2] about designing with resets in digital designs. These publication consider the advantages and disadvantages in using synchronous vs asynchronous resets of the flip flops of an RTL design. They also suggest methods to overcome any disadvantage to produce a more reliable reset network. The first work [1] strongly asserted that asynchronous were the best choice, but after critical review, the second paper [2] held a more passive view and stated both resets could be effective in digital design.

The author of this paper believes that those publications provided a good general view of resets in digital designs for both ASICs and FPGAs. The main focus of the author's work is designing with Xilinx SRAM-based FPGAs. There is a lot more architecture specific details to considering when design resets on these platforms. After reviewing suggested additional reading from Chu's textbook [3] ,and Xilinx forums [4], [5] and white papers [6], [7], the author believes the advantages of asynchronous resets listed in the previous works are not effective. Xilinx FPGAs have native power-on resets, flip flop primitives that support both types of resets and target active high resets, and a limited global reset network. These features are critical when design resets within a Xilinx FPGA. The author supports the opinion of Chu and Xilinx Engineers, that the power-on reset acts as a initial global reset to set the device into a known state and that any other required reset should be a local synchronous solution.

II. Resets

Resets are critical in digital design and may be required for simulation and final implementation. Resets set the design in to a known working state for both simulation and final implementation. ASICs may require a reset on power up to prevent any state logic starting in a bad state [2].

A. Synchronous Resets

A synchronous reset only affects the digital design on a chosen clock edge. Synchronise resets insure that the system is completely synchronous and no glitches on the reset lines will disrupt the logic. Additional flip flops could be used to extend the reset buffer tree to help maintain timing over the whole device. One of the reviewed publication [2] states that dependent on the synthesis tools, the reset can be applied to the flip flop as part of the combinational logic for the d-input. This assumes that the flip flop does not have a dedicated synchronous reset port. Not all ASIC libraries support flip flops with built-in synchronous resets and synthesis tools may not easily distinguish the reset signal from any other data signal. A synchronous reset will be synchronised to one clock domain and may have to be widened to work with a multi-clock design. This clock domain must be active on power-on to set the whole device into a known state. This may be difficult to do with some designs with gated clocks.

B. Asynchronous Resets

Asynchronous resets allows the designs to be set into a know state at any time without worrying about any additional clocks, signals, or state. The biggest issue of these resets is the asynchronous release of the reset putting any flip flops into metastability. Many ASIC libraries include flip flops with a asynchronous reset pin for easy implementation. With these libraries, there is no additional delay added to the reset line and improving the overall timing of the design. The publication [2] states that many of the common issues stated for asynchronous reset are concerns that are not true, such as incorporation into a cycle based simulator, or problems that can easily be solved, such as determining static timing. This publication also purposed an easy solution to the "biggest" issue of the asynchronous release by synchronising the release of the reset with a simple circuit. This allows for an asynchronous reset to be synchronise on the release allowing the designer to have the advantages of both circuits.

III. Reset Solutions

The reviewed publication suggested many solutions to improve the reliability and timing of the reset networks. Their first solution was a method to synchronize the release of the synchronous reset. The method for this was to use a set of flip flops (example of two) to drive the reset for the entire device as shown in Figure 1. The D input of these shift registers was connected to a logical one to emulate a de-asserted reset. The true reset line was connected to the reset pins of these flip flops

so if a reset was asserted, the entire shift register would be set to logical zero issuing a reset through the entire device. When the reset was de-asserted, the shift registers would provide a synchronous cycle delayed de-assertion to the entire network. This solution solved the "biggest" issue of the asynchronous releases, but did not prevent glitching on the reset line from issuing an erroneous reset.

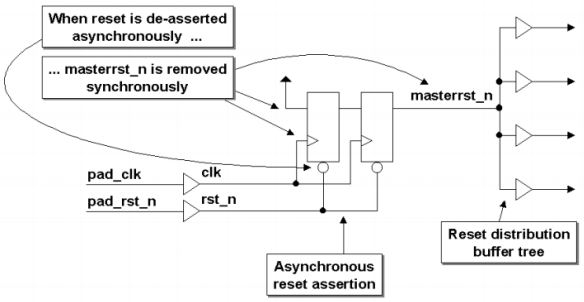

Fig. 1. Reset Synchronizer [2]

Another issue in digital design is distributing to all the filp flops within a given time window. Like clock distribution , the reset can be distributed as well. These papers [1], [2] proposed two solutions. For synchronous solution, simple flip flops can be added to reset line to decrease the delay path from any reset to any data flip flop. A similar solution exists for asynchronous reset, instead of using a flip flop, additional synchronizing circuits (Figure 1) can be added to the design to decrease the delay path of the de-asseration of the reset as shown in figure 2. This is only a concern with a high fanout global reset which your design may not require.

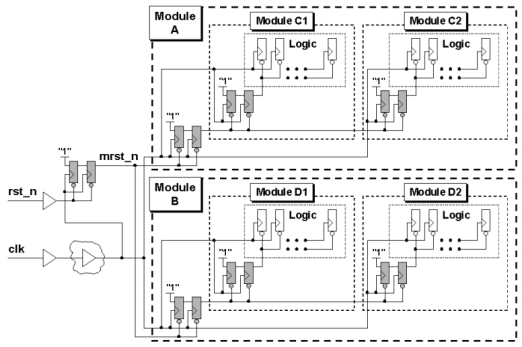

Fig. 2. Asynchronous Reset Distribution Tree [2]

Another solution presented was a simple circuit to prevent glitching on the asynchronous resets. This solution compared the reset input to a delayed reset signal to verify that the reset was stable for a given delay before issuing it to the reset of the device. This is similar to a simple button de-bouncer to prevent noise on the line from being registered by the system. Though this solution is simple, it provides an effective glitch, though delayed, reset.

IV. Vendor Tools

The reviewed publications made strong assumptions about the the vendor library, and simulation and synthesis tools. Many of the disadvantages listed in the work can be associated with these aspects and not the nature of the resets. Vendor libraries may support different types of flip flops. Issues in simulation and synthesis may be solved by using different or newer tools. These tools may have the support to properly simulate and implement both types of resets. An update from the current development tools and available vendor libraries would greater improve these publications.

V. ASICs

The reviewed publications [1], [2] focused mainly on ASIC designs. The author of this review assignment does not have any experience or background to refute their claims and overall agrees with their statements concerning ASIC design. ASICs have design libraries when with specific flip flops and may not have the flexibility that is built into FPGAs. In ASIC design, each transistor counts and even a synchronise reset may result in more power consumption. ASICs may have the requirement of a power-on asynchronous reset before a clock is available. This is dependent on the technology, requirements, and available libraries for the specific ASIC design.

VI. Xilinx FPGAs

FPGAs differ from ASICS in that the routing and transistors are already established. The selected textbook [3] stated that asynchronous resets should be used for system initialization and not during run time to clear any registers. This section will introduce the flip flop primitives on the FPGA, how the synthesis tool applies resets, the guaranteed known state at power-up, and the use of local resets. These topics will show that the assumptions from the reviewed publication are not true in FPGA designs.

A. Flip Flop Primitives

The FPGA only has one type of physical flip flop with defined ports, but it can be configured into several different types of flip flops during synthesis. These are a few of the configuration that are available for the flip flops. The flip flops can be configured with a asynchronous clear setting the registered value to 0, an FDC (Figure 3). The flip flop can also be configured so that the asynchronous signal sets the registered value to 1, an FDP (Figure 4). This matches what the reviewed publications where expecting in the ASIC primitive library.

In addition to the the asynchronous options, each flip flop can be configured with synchronous reset ports. This sounded abnormal in the reviewed publications as ASIC libraries may prefer asynchronous resetable flip flops. There is no observable difference in power or performance in using the different ports on a Xilinx FPGA flip flop in the Xilinx documentation [8]. The flip flop can be configured with a synchronise reset , an FDR (Figure 5), or a synchronise set, an FDS (Figure 6).

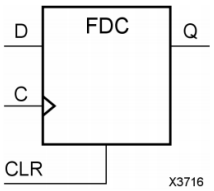

Fig. 3. D Flip-FLop with Asynchronous Clear [8]

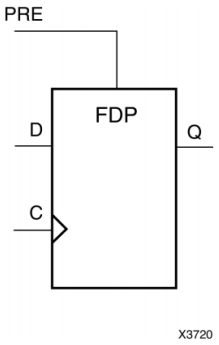

Fig. 4. D Flip-FLop with Asynchronous Preset [8]

It is important to not mix asynchronous and synchronous types when inferring the flip flops. According to Xilinx documentation [7], when the user defines multiple controls for a flip flop, the synthesis tools must add additional combinational logic to satisfy the HDL design. This can greatly increase the utilization of the selected design requiring more power, better speed grade, or a bigger part.

B. Vivado Synthesis

The authors of the reviewed works [1], [2] use an out of date version of Synopsys to generate their schematics. The synthesis is important in considering how the reset network is created and different tools produce different results. The authors did note that some tools may have difficulty identifying resets from other D inputs. The author of this paper compared the synthesis results of Vivado to the ones presented in the reviewed publications. One big difference was the synchronized reset for the flip flops. The reviwed worked presented Figure 7 as the synthesised results of Listing 1. A and gate is used on the d input of the first flip flop to emulate a synchronise reset.

The Listing 1 was synthesized using Vivado 2019.1 with the reset being changed from active high to active low. According to Xilinx documentation [8] and forums [5], all the flips flops use active high controls for the resets. The insertion of any active low controls will add logic to the reset path. Figure 8 shows the reset line going directly to the flip flop without any additional combinational logic. The Xilinx primitive library has flip flops that can be implemented with synchronous or asynchronous controls without any additional combinational logic needed.

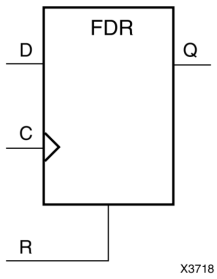

Fig. 5. D Flip-FLop with Synchronous Reset [8]

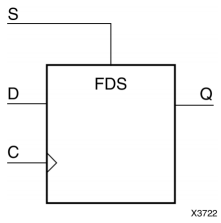

Fig. 6. D Flip-FLop with Synchronous Set [8]

C. Power-on Reset

One major concern of the reviewed publications [1], [2] and the textbook [3] was ability to reset the device to a known state on power-up. This may be impossible for a synchronous reset and may require a asynchronous solution for an ASIC device. According to a Xilinx white paper [6], When an FPGA is configured or reconfigured, every cell is initialized. This configuration as the exact same effect as a global reset for flip flops and in addition, initializes all the RAM cells. With this feature, the FPGA is in a ideal state to match simulation and remove any need for a boot-up sequence to configure the RAM memory. This is a native power-on reset feature that removes any needs for a global reset to span the entire device.

D. Local Resets When Needed

Global resets can be harmful to an Xilinx FPGA designs. The same white paper [6], states that global resets can use up routing and logical resources and prevent the use of highly efficient features, such at the SRL16E. When resets are needed in the design, the white paper suggest using high-performance localized synchronous resets with low fan-out instead of a large reset distribution tree. The flip flops being reset can use the synchronous set (FDS) or reset (FDR) to implement a fully synchronous design, and easy timing specifications and analysis.

E. Asynchronous Glitching

Using asynchronous reset without a anti-glitch circuit, as a global reset on FPGA may lead to unreliable sporadic behaviour as reported by this Xilinx forum [4]. The reset trace on the PCB can be susceptible to glitching and thus resetting all the flip flops within the FPGA. This issue can be hard to debug and as shown in this paper may not be required for the FPGA design. Relying on the FPGA power-on reset or a

Listing 1. Good VHDL coding style to model dissimilar flip-flops [2]

```
library ieee;
use ieee.std_logic_1164.all;
entity goodFFstyle is
    port (
    clk : in std_logic;
    rst_n : in std_logic;
    d : in std_logic;
    q2 : out std_logic);
end goodFFstyle;
architecture rtl of goodFFstyle is
    signal q1 : std_logic;
begin
    process (clk)
    begin
        if (clk'event and clk = '1') then
        if (rst_n = '0') then
            q1 <= '0';
        else
            q1 <= d;
        end if;
        end if;
    end process;
    process (clk)
    begin
        if (clk'event and clk = '1') then
            q2 <= q1;
        end if;
    end process;
end rtl;
```

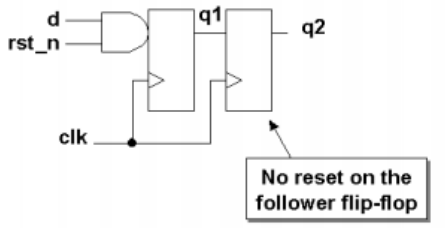

Fig. 7. Two different types of flip-flops, one with synchronous reset and one without [2]

synchronise reset can prevent glitches on your IO lines from creating hard to debug problems.

VII. Conclusion

The reviewed worked targeted a general audience and provided good information to considering when developing a RTL design. It is important to understand the specifics of architecture that is being developed on. ASICs and FPGAs share similar development tools such as HDL languages and synthesis tools, but require different techniques to optimize the design. The tools present are outdated and new tools may provide better solutions. The reviewed publications focused on

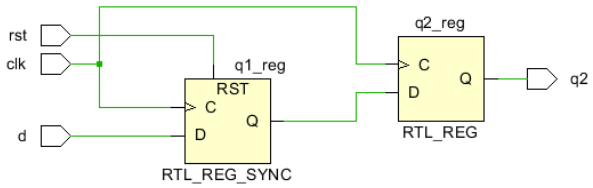

Fig. 8. Flip flop with active high Synchronous reset

ASIC design and lack important details for FPGA developers. This is completely understandable, as FPGA technology varies between companions and products within companions.

Xilinx FPGAs have native power-on resets, flip flop primitives that support both types of resets and target active high resets, and a limited global reset network. The author supports the opinion of Chu and Xilinx Engineers, that the power-on reset acts as a initial global reset to set the device into a known state and that any other required reset should be a local synchronous solution.

References

[1] C. E. Cummings, "Simulation and synthesis techniques for asynchronous fifo design," in *SNUG 2002 (Synopsys Users Group Conference, San Jose, CA, 2002) User Papers*, 2002.

[2] C. E. Cummings, D. Mills, and S. Golson, "Asynchronous & synchronous reset design techniques-part deux," *SNUG Boston*, vol. 9, 2003.

[3] P. P. Chu, *Sequential Circuit Design: Principle*. IEEE, 2006. [Online]. Available: https://ieeexplore.ieee.org/document/5237649

[4] Chapman, "That dangerous asynchronous reset!" May 2013. [Online]. Available: https://forums.xilinx.com/t5/PLD-Blog-Archived/That-Dangerous-Asynchronous-Reset/ba-p/12856

[5] Balacha, "Demystifying resets: Synchronous, asynchronous other design considerations... part 1," Jun 2019. [Online]. Available: https://forums.xilinx.com/t5/Adaptable-Advantage-Blog/Demystifying-Resets-Synchronous-Asynchronous-other-Design/ba-p/882252

[6] "Get smart about reset: Think local, not global." [Online]. Available: https://www.xilinx.com/support/documentation/white_papers/wp272.pdf

[7] "Get your priorities right – make your design up to 50% smaller." [Online]. Available: https://www.xilinx.com/support/documentation/white_papers/wp275.pdf

[8] "Xilinx 7 series fpga libraries guide for schematic designs." [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/7series_scm.pdf