

ASYNCHRONOUS FPGA RISKS

Ken Erickson
Jet Propulsion Laboratory, Pasadena, CA 91109

Abstract

Synchronous versus Asynchronous Logic

A logic designer often has the choice of implementing FPGA logic using synchronous or asynchronous logic structures. All flip-flops in synchronous logic are clocked using the same global clock. When this is not the case, the design is asynchronous. The following are examples of asynchronous logic:

1. All flip-flops are not clocked using the same global clock.
2. The output of a logic gate or another flip-flop is used as a flip-flop clock.
3. The output of a logic gate or flip-flop is used as a flip-flop set or reset signal to control flip-flop state, except for the system reset function.

This synchronous design can be easily analyzed using available design tools to verify that the logic performs the intended function and that flip-flop setup and hold times are met under worst case condition. In other words, it can easily be verified that the design is bullet proof.

However, when a designer chooses to use an asynchronous logic structure, it may be very difficult to verify that the design meets performance requirements under worst case conditions. The reason for this, is that FPGA design tools are not tailored to perform worst case timing analyses of a logic structure having multiple clocks. As the result, the designer will be required to manually check the worst case timing of each clock and data path to verify that worst case setup and hold time requirements are met for all flip-flops. For complex asynchronous logic structures, this process may be extremely tedious or next to impossible. The end result may be a design having unknown setup and hold time margins as well as a design that has the risk of failing due to timing race conditions.

Risk of Asynchronous Logic

Assume that an asynchronous FPGA having unknown timing margins is used in a spacecraft application. During the mission, there is the risk that the assembly using the FPGA will not perform its intended function during the space mission as the result of FPGA timing variations due to aging, radiation and temperature. If the FPGA is used in a critical assembly, a failure of the FPGA to perform its intended function could result in the end of the mission. Even if the FPGA is used an assembly which has a redundant counterpart, the same problem or similar problems can result in both units.

Risk Assessment

Asynchronous FPGA risks can be assessed by using one or more of the following methods:

1. Detailed review of the FPGA logic design documentation (e.g. HDL documentation, schematic) by a logic design expert to identify potential problem areas.
2. Worst case timing analysis of potential problem areas identified by the reviewer.
3. Voltage/temperature margin testing (small temperature steps preferred).

Keep in mind that the risk is based not only on the likelihood of the FPGA failing to perform its intended but is also based on the impact this failure has. Consider an FPGA which is used in a spacecraft assembly. If a failure of the FPGA would result an end to the mission, the risk would than it would be if the FPGA failure did not result in an end to the mission. For example, the risk of FPGA logic in a Star Tracker would probably be higher than the risk of the same FPGA logic in an instrument.

Risk Mitigation

Once the risk is assessed, the decision must be made on whether or not the risk should be partially or completely mitigated. Possible risk mitigation options are as follows:

1. Redesign logic to make it a synchronous design. This is the preferred option if it not precluded by budget or schedule constraints. A timing analysis of the new design can then easily be performed using available FPGA design tools to verify that worst case timing requirements are met. This is the preferred mitigation method.
2. Perform complete timing analyses of the design. This may be next to impossible if the design is complex and may take more time and be more costly than a complete redesign.
3. Modify selected portions of the design which have been identified as having the highest risks. A worst case timing of analysis of the modified logic should then be performed to verify that worst case timing requirements are met. This action only partially mitigates the risk.
4. Perform voltage/temperature margin testing. In this case the assembly using the FPGA is tested over temperature at different supply voltages. Logic propagation delays are affected by both supply voltage and temperature. Very small temperature steps (such as 1 degree centigrade) are recommended since timing problems may occur over a narrow temperature range. During the testing, some FPGA timing problems may surface. These problems can then be corrected. Even if no problems are found during the testing, the design confidence is increased somewhat, but not completely. Aging and radiation can cause some delays to increase and others to decrease in the same FPGA. Voltage/temperature margin testing will not simulate these affects. For this reason, this action only partially mitigates the risk.
5. Prayer.

Recommendations for the Future

The following actions can be used to virtually eliminate the asynchronous design risks:

1. Adopt company and/or project policy that all designs be synchronous except for special circumstances where an asynchronous design is justified (possibly for speed or power dissipation reasons, for example).
2. Adopt the company and/or project policy that requires that complete worst case timing analyses will be performed on all logic designs to verify that adequate timing margin exists.
3. Adopt company and/or project policy that requires that each logic design and timing analysis be reviewed by a peer who is an expert logic designer. This is important even for synchronous designs since subtle problems can still exist. For example, logic state machines may need to be initialized properly or the logic may fail to function as required. A synchronous reset logic may be required for proper initialization. An expert reviewer is more likely to identify this potential problem.