How Much Insurance Can You Afford?

Bruce Cory
NVIDIA
2701 San Tomas Expressway
Santa Clara, CA 95050
bcory@nvidia.com

Abstract
Most chips designed have debug logic. Debug techniques including visibility of a debug bus, controls for complicated circuitry like RAM self timing or high speed I/O, and clock control have been used for many years by many companies. Many older techniques like scan enabled debug and on-chip logic analyzers and emerging techniques like specialized on-chip instrumentation look interesting but are not used in most designs today. The big question is why don’t some of these techniques catch on more and how much debug logic is needed to mitigate the risk of slow time to market?

1 Panel Overview
To answer the question of why some techniques do not gain wider acceptance, debug solutions must be divided into groups that represent the most used and non-used techniques and scrutinized to identify the differences in the groups. Three groups are described below and several reasons are given for why these groups have more or less widespread use. Since NVIDIA is a chip design company that uses almost no third party IP solutions, the below thesis is more applicable to applications in that space.

2 Panel Thesis
There are several possible techniques to enable debug. I will divide them into three main groups. The first group includes debug techniques that primarily use logic that is needed on the device for other reasons. For example, most of the hardware area needed for scan logic dumping is already needed for scan testing. These solutions can usually be implemented and verified by the group that needs the logic for the non-debug reason at a reasonable extra cost. The second group includes solutions that are very easily added to an existing piece of logic with almost no overhead cost. A good example of this is memory self timing controls. The memory designer can own implementation as it doesn’t add much design area or complexity above what is already needed. The third group includes all other techniques. A good example of a solution in this third group is on-chip instrumentation. These don’t have the ability to leverage hardware that is primarily used for other reasons and do not meet the no-overhead/easily implemented criteria. My position is that the techniques that fall into the third group are very hard to get implemented on a design while groups one and two are more widely adopted and used.

The cause of the above thesis is due to several reasons. The primary one is that debug logic is mitigation of slow time to market risk for a device. When that logic isn’t leveraged by other needs, justification purely using mitigation of risk is hard. Usually this requires a company that has been stung by previous design issues or an application space where functional issues must always be root caused and understood. A secondary reason is resource cost. One of the main costs of implementing on-chip debug logic is the repeated verification of that logic while the chip is going through timing closure and the final ECOs. This cost can overshadow more obvious costs like area overhead. After fighting through the justification and the resource issues, the design teams must also be willing to handle the fallout if the work done does not prove useful from a ROI standpoint.

3 Panel Summary
Three different groups of debug solutions have been presented along with practical reasons that mostly dictate the use of these solutions. Debug solutions are very different from functional logic or test logic because the primary goal is risk mitigation of slow time to market.