Thoughts on Design Reviews

To preface these thoughts, let me first explain their source and basis. During my early career, I was fortunate enough to work in the commercial aircraft industry as it was starting to grow, then to be involved in various projects from earth centric satellite work and the first lunar landings as part of the Apollo program to the first successful working probes to land on Mars with the Viking Landers as part of the early space program. Later work entailed very high quality, high reliability, test equipment and state of the art medical devices. A common thread though all of my work has been the need for high reliability and the recognition that failure could easily cost people their lives or significant amounts of money. A service pack could not mitigate the consequences of a failure on an Apollo capsule bound for the moon, a Viking Lander with a failed comms link millions of miles from earth, a multimillion dollar weather satellite in the wrong orbit, or in piece of medical equipment failing while being used on a critically ill patient. While certainly not every project possesses failure modes with such extreme consequences, approaching every design as if it did can only help to make that design better. It is this background that has molded my philosophy and approach to design today.

An important component in the development life cycle in all of the cases I have noted was a series of critical reviews of the thing being designed. The few lines that follow are not a how-to check list for conducting such reviews or ensuring perfect code. Rather, the objective is to stress the importance of such practices and to suggest some high level guidelines.

Most formalized systems development processes recognize that the cost of correcting a fault increases as progresses through the process. We often refer to this as the hockey stick curve because of its shape resembling a hockey stick. It is typically found that effort spent during the early stages of development to anticipate, discover, and correct errors has very positive effects at the end of the day.

The objective of reviews throughout the development cycle should be to try to find and identify design weaknesses or errors, oversights, or potential fault modes. Such an exercise can help to ensure (it is absolutely not a guarantee) that the design meets all of the original requirements, is solid, robust, reliable, safe, and when it fails it fails in a graceful manner. At the same time, we

also need to recognize that physical things do fail. Thus, we want to execute our designs to mitigate the effects of such failures as much as possible when they do occur.

As with any of the other activities supporting the development process, design reviews, code annotation and documentation, or test procedures are not the end deliverable. Conducting a design review every few days or producing volumes of documentation can be as counterproductive as never holding one or never documenting your work. Conducting a 'design review' simply be able to say, 'well, that's done', to meet a milestone that appears on the project schedule is of little use.

Throughout the development life cycle, a key element of any design review process is what we call *egoless design*. One cannot let the belief that the best widget ever witnessed by humankind is in the process of or has just been designed to get in the way of an unbiased assessment of that design to determine if it is really the case. It is absolutely essential that ego not be a part of the review the process.

Design reviews, as well as supporting code walk-throughs and code inspections, are most effective if done by the fresh and unbiased eyes of someone or some group not directly involved with the project. As is often the case when proofreading one's own writing, schematics, or code, our brain ensures that they appear exactly the way we want them to rather than as it's actually written or designed.

Following the five steps to design outlined in the Lesson 1, a good first review ideally should follow the writing of the requirements and system specifications. Such a review can ensure that everyone understands the high-level specifications and functionality of the system before proceeding. Thereafter, a design review before the architectural phase can confirm detailed functionality and the communication links between and amongst functions. As the design progresses, at least one review prior to moving to prototype development can help to ensure that the partitioning and mapping of the software from function to processors, FPGAs, or ASICs, is sound. The formality of such reviews can vary with need and with project. They can range from a simple exchange of documentation and code amongst the team members to detailed reviews with reviewers who are not directly involved with the project.

No matter how one chooses to proceed, it is important that the review be conducted in a positive and constructive manner. Ridicule, blame, deprecation, or sarcasm have no place in a professional development or review process. All participants should recognize that a good, cooperative, review helps everyone to ensure a higher quality product at the end of the day.

Returning to egoless design for a moment before closing. Good designers and developers are rightly proud of their work. This is only natural. Hearing that someone has found a problem with a portion of your design is never a pleasant thing to hear. The discovery, discussion, and correction of such problems should always be conducted in a positive and constructive manner and should (sure, perhaps reluctantly some times) be received with thanks.