Using Hazard-Based Engineering Standards to Guide Software Design

Anura Fernando
Underwriters Laboratories, Inc.
333 Pfingsten Road, Northbrook, Illinois 60062

Abstract. Risk Management is the process of reducing the likelihood of hazard manifestation. Third-party test organizations have the unique advantage of historical familiarity with a plethora of solutions for a given problem and the field histories associated with implementation of these solutions. This knowledge can be applied, generically, when developing Standards related to the risk management of such design solutions. Regulatory issues are frequently misunderstood and ignored until final phases of the development life-cycle. We will examine how industry can make optimal use of these Standards by utilizing them for requirements development and hazard mitigation during early phases of software design.

BACKGROUND AND SCOPE

General. In order to understand how to optimise a process, we must first understand the process itself. We will gain an understanding of how best to utilize Standards by examining design, root-causes of system failure, and mitigation of hazards that may manifest as a result of failure (fault).

Definitions. Hazard and Risk are, as with most abstract concepts, a matter of perspective. Detonation of an explosive weapon, for instance, can be desirable or not, entirely dependent upon one's frame of reference. Thus, in a general sense, a Hazard is an undesirable event, and Risk is the assessment of probability and severity associated with the manifestation of the Hazard (IEC, 2000).

Scope. Due to the necessary ambiguity of these definitions, we will constrain this discussion to the domain of Programmable Electronic Subsystems (PESS) and the risks of fire, electric shock, and casualty to persons. Although we will focus on the example Standard ANSI/UL 1998 - Software in Programmable Components, the fundamental concepts may be applied to other Standards, other Hazard domains, and other Risks.

STANDARDS DEVELOPMENT

History. Standardization has likely existed since people developed the cognitive ability to recognize objects as "similar" and "different." Our sample Standard, ANSI/UL 1998 was developed by Underwriters Laboratories, which has been involved with the development of UL, ASTM (American Society for Testing and Materials), IEC (International Electrotechnical Commission), and ISO (International Standardization Organization) Standards since 1894. During the ensuing one-hundred-plus years, UL has worked with other organizations and industry to develop more than 1500 Standards, with continual updating and/or obsolescence as technology evolves (Bezane, 1994).

Foundations. Some of the earliest research conducted for hazard identification and risk determination related to electrical fires began in the 1890's as fires began to plague American cities. A member of the National Board of Fire Underwriters was quoted during that time as saying, "Better buildings are burning in a greater ratio than ever before…and there are mysterious causes at work that we do not understand. I believe (the cause) to be electricity" (Bezane, 1994).

In a similar vein, "electric shock" research has been, and still is progressing, due to contributions to Standards by collaborative efforts between Standard developers and academic researchers. Such collaboration is evident in the contributions of the Applied Physics Laboratory at Johns Hopkins University, the Physiological Institute at the University of Freiburg, the Department of Neurosurgery at the Medical College of Wisconsin, and the College of Engineering and Applied Science at Arizona State University to the basic science supporting Standards' requirements regarding limits for electric current through the human body, leakage current/voltage criteria, and design considerations such as grounding, double insulation, ground-fault-circuit interruption, shielding, and polarization (Reilly, 1992).

USING STANDARDS TO IDENTIFY HAZARDS

Casualty. The hazard domains of "fire" and "electric shock" tend to behave within the constraints of the physical sciences, but casualty encroaches the bounds

of imagination. Casualty considerations move from the realm of defined parameters to that of functionality and intended use. Thus, the first step in addressing casualty concerns is to define the hazards and conduct a risk analysis. It is during this initial step that a well-informed systems engineer can begin utilizing Standards to assist with the development process. Two such useful Standards are EN/ISO 14971 "Medical Devices – Application of risk management to medical devices," and EN 1441 "Medical devices – Risk Analysis." Both of these are products of the International Electrotechnical Commission's collaborative efforts with other organizations such as the International Organization for Standardization and they are adopted as European Norm Standards. The hazards defined by using such Standards can be included, along with such items as customer needs, intended uses and available technologies, as inputs for Requirements specification. However, not all hazards can be eliminated by design and thus remain inherent (residual) to some extent. This residual risk must be assessed and accepted or mitigated before further development.

RISK ANALYSIS

Approach. Several techniques exist for conducting risk analysis, and one of the tenets of our sample Standard, ANSI/UL 1998, is that neither analysis nor design are to be dictated by the Standard contingent upon satisfaction of its basic requirements with respect to safety.

Techniques. Using a checklist is one approach to risk analysis, but relies on trial and error history and corporate memory to generate a sufficiently comprehensive list. Checklists guide thinking and can be used in all life-cycle phases, but can be problematic if used without careful consideration of each item on the list (Leveson, 1995).

Hazard indices can also be used for risk analysis by measuring potential loss. This technique works well within domains with little variance between system designs, but is not conducive to systems with a variety of design options or rapid changes in technology (Leveson, 1995).

Fault Tree Analysis (FTA) is a means of identifying potential causes of hazards. It consists of system definition, fault tree construction, qualitative analysis, and quantitative analysis. It requires foreknowledge of the system, and caution must be excercised to prevent oversight of critical paths due to oversimplification of system representation (Leveson, 1995).

Cause-Consequence Analysis (CCA) starts with a "critical event" and uses a top-down approach or backward search to determine the cause of the event and the potential consequences of the event occurance. CCA diagraming can become unwieldy due to the need for an individual diagram for each initiating event (Leveson, 1995).

Hazard and Operability Analysis (HAZOP) is a qualitative technique used for the identification of deviation from expected operation and the associated hazards. If conducted under ideal circumstances, this technique can identify and/or eliminate a great number of hazards. However, the "ideal circumstances" rely heavily on the experience and judgement of the engineers performing the analysis (Leveson, 1995).

Failure Modes, Effects, and Criticality Analysis is useful for discrete failures. It can be used to establish the overall probability that the product will operate without a failure for a specific length of time or for a specific length of time between failures. This technique, although comprehensive, can be burdonsome because of the need to exercise each failure mode of the device under evaluation (Leveson, 1995).

SYSTEM FAILURE MODES AND MODELLING

General. In order to effectively pursue a discussion of system failure analysis and minimize divergence into various domain backgrounds, we will constrain the scope of this discussion to include software- and microelectronic-based failures of Embedded Systems.

Software. The consensus, in Software Engineering, over the past few decades has been that "software does not fail," and rather programmers have either been mislead with incorrect specifications according to which to develop the software, or the software has been incorrectly implemented.

Current software Standards, such as our example ANSI/UL 1998 (Software in Programmable Components), strive to encompass both 'implementation' and 'non-implementation' defects in software. Awareness of and use of such Standards early in the software development life cycle can lead to improved development by such mechanisms as enhanced static testing (ie. Code Walkthroughs, Code Inspections, etc…), dynamic testing (ie. Boundary Condition, Range Testing, etc…), and formal testing (ie. Mathematical Proof, Control Modelling, etc…) (IEEE, 1993).

Static testing can be used to uncover such defects as have occurred in NASA's Mars Climate Orbiter, where miscalculations of thrust for trajectory adjustment were

attributed to inconsistency in measurement units, with some (ground) software using metric units of impulse (Newton-seconds) while ‘in-flight’ output was specified in English units (pound-seconds) (Fusco, 2000). Other defects potentially detected by static testing include: algorithm/logic/processing defects (ie. ‘off-by-one,’ return codes, overflow/underflow, etc.), data defects (ie. Pointer errors, indexing, initialisation, etc), and system errors (ie. Stack control, version control, resource sharing, etc.) (Beatty, 2000).

Dynamic testing, within the context of a structured development life-cycle, can be used to expose defects similar, in nature, to those seen in the case when, in 1988, the USS Vincennes’ Aegis radar system software operated as intended with respect to aircraft identification of an Iranian airliner, but was functionally deficient with respect to the GUI or human factors design; leading to the loss of the lives of all 290 passengers onboard the airliner (Gannsle, 2002). Other defects that may be exposed by this type of testing include: incorrect control flow, re-entrance errors, data synchronization errors, task synchronization errors, instrumentation problems, etc (Beatty, 2000).

The types of defects that may be exposed by formal testing parallel and, in some instances, overlap those of the previous two testing methodologies. Formal testing can be used to demonstrate via Formal Methods (ie. Topology and Set Theory) that all Requirements have been implemented, it can be used to demonstrate System stability, and it can be used to demonstrate correctness and completeness of algorithms.

Microelectronics. The second half of the ANSI/UL 1998 approach to establishing a baseline of Embedded System safety is the examination of microelectronic root causes of ‘software failure.’ Clause 8 of the Standard provides for, “Analysis of possible combinations of microelectronic hardware failures, software faults, and other events that are capable of resulting in a risk. This includes, for example, microelectronic hardware failures that cause software faults that are capable of resulting in a risk.” In order to begin analysis of such failures, it is necessary to gain a systemic understanding of the impact of microelectronic hardware upon the software.

The most prevalent technology in the embedded processor industry is currently CMOS (Complementary Metal Oxide Semiconductor). Thus, microelectronics-based ‘software’ errors should be examined from the very foundations of the technology, since ‘software’ is essentially the manipulation of 1’s and 0’s or control of electron densities within the crystal lattice structures of semiconductor materials.

1’s and 0’s. As we have alluded, ‘software’ is nothing more than the instantiation of thought, implemented in binary logic with the behavioural constraints of Boolean Algebra. In the field of Programmable Systems, we have gone from the likes of the ENIAC computer, of the 1940’s, requiring 18,000 vacuum tubes for binary logic implementation to the Pentium 4 computer, of the new millennium, with 45 million transistors for binary logic implementation (Gannsle, 2002).

Currently, the fundamental microelectronic unit of CMOS-dependent ‘software’ is the CMOS transistor, which is capable of representing a logical condition of ‘0’ or ‘1.’

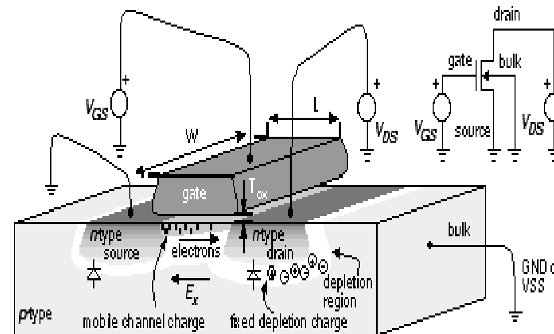

Figure 1. n-Channel CMOS Transistor (Smith, 1997)

The CMOS transistor is a four terminal device that can be ‘turned on’ or ‘turned off’ using the ‘gate’ terminal. An n-channel transistor requires a logic ‘1’ on the gate to make the ‘switch’ conducting, and a p-channel transistor requires a logic ‘0’ to make the ‘switch’ conducting. Connecting an n-channel transistor in series with a p-channel transistor produces an ‘inverter.’ Four such transistors can be used to form a ‘two-input NAND gate’ or a ‘two-input NOR gate’ (Smith, 1997).

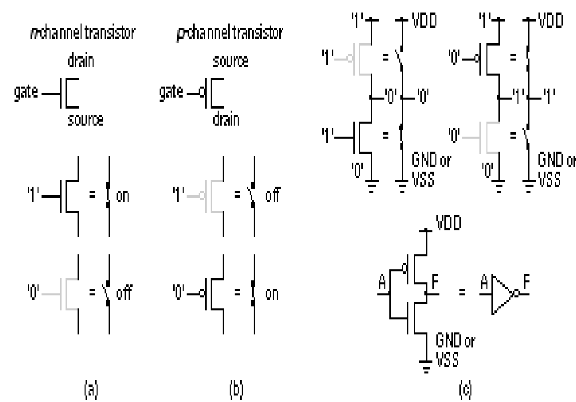

Figure 2. Basic CMOS Logic (Smith, 1997)

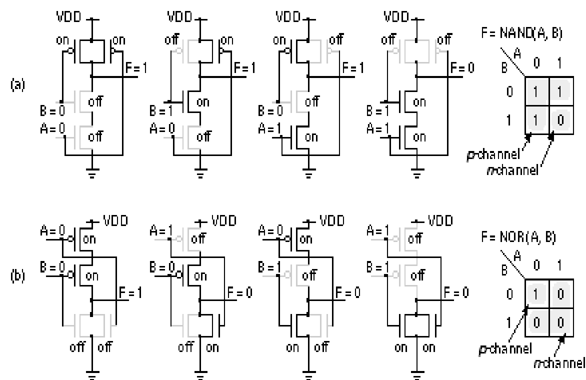

Figure 3. CMOS Gates (Smith, 1997)

The development of this hierarchy of increasing complexity adds additional dimensions to understanding the impact of microelectronic failure modes on the 'software.' Reducing the degree of abstraction in Figure 3, we see that Boolean Operators, in application, are subject to phenomena of semiconductor physics that may not be represented in the mathematical models used in design.

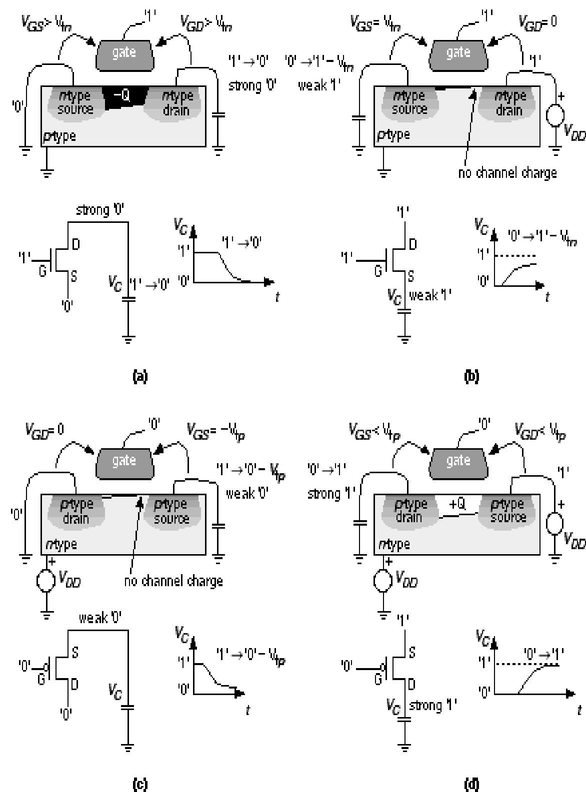

Figure 4. The Physics of Logic (Smith, 1997)

Thus, we have the basic ingredients for implementing 'software,' as well as one of the first potential hazards: logic errors within a critical functional thread.

Memory. In addition to manipulating 1's and 0's, 'software' can be much more versatile, if we are able to store discrete 1's or 0's as 'bits' or if we are able to store groups of 1's and 0's as 'bytes.' We previously established that 'software' is an extension of pure thought or ideas. We now see the possibility to develop electronic 'cognition,' since 'memory' implies 'remembering,' 'learning,' and possibly even 'understanding.'

CMOS technology allows for the most common types of memory architectures: 'volatile' memory and 'non-volatile' memory. Memory is frequently arranged in a grid of 'memory cells' in order to allow for the unique identification, relative association, and retrieval of each piece of information.

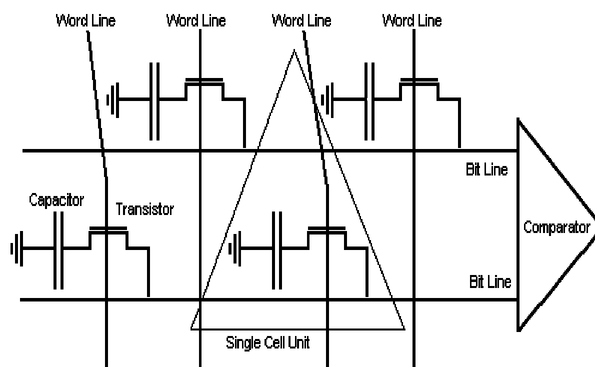

Figure 5. Example Memory Array – DRAM (Suzor, 2002)

Memory cells, when reduced by one order of abstraction, as in Figures 6 and 7, can be easily seen to be susceptible to many of the same failure (fault) mechanisms inherent in CMOS logic architectures.

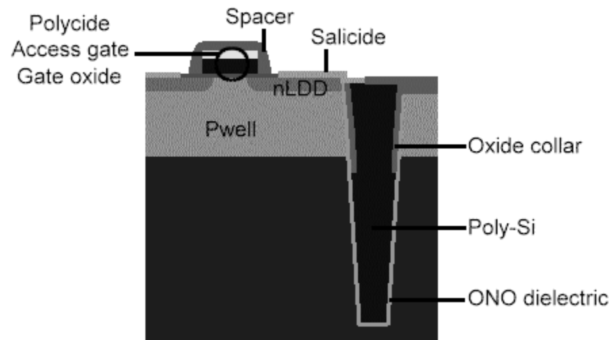

Figure 6. Trench DRAM (Suzor, 2002)

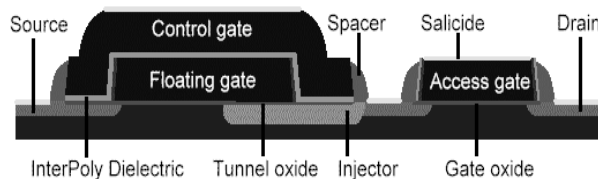

Figure 7. Type 1 EEPROM (Suzor, 2002)

Fabrication. The first source of potential microelectronic failures in CMOS devices is during the IC fabrication. In application, ANSI/UL 1998 addresses this type of failure by reliance on the endproduct and reliability Standards' requirements for 100% production-line testing of safety-critical functions. However, to understand the failure mechanisms, we must examine the fabrication process. The fabrication of integrated circuits is a very complex process, well beyond the scope of this discussion. However, to understand the potential failure modes of Embedded Systems, a summary review of The CMOS Process (Smith, 1997) will be necessary. Figure 8 provides an overview of the general process steps in CMOS fabrication, beginning with a single-crystal silicon ingot (boule), and ending with a functional circuit.

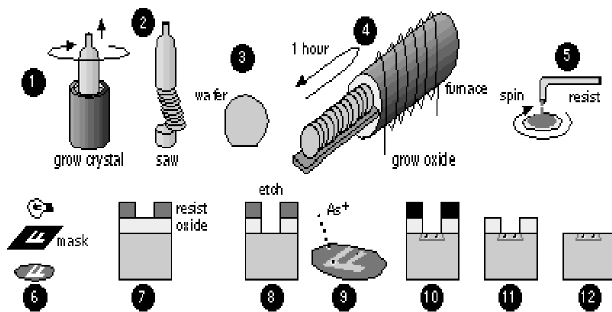

Figure 8. The CMOS Process (Smith, 1997)

The boule is drawn from a crucible containing a melt that is maintained at 1500 °C, slightly higher than the melting point of Silicon. Acceptor or donor dopants are introduced into the melt to create p-type or n-type silicon, respectively. The boule is sectioned into circular wafers between 6 and 12 inches in diameter and typically approximately 600 micrometers thick. A batch of wafers is placed on a 'boat,' which in the environment of a furnace will grow a layer of SiO_2 that provides the chemical and electrical properties for controlled conduction (ie. Switching). Finally, a series of masking / etching steps (with multiple substeps) creates the layers that define the 'components' and the metal interconnects (Smith, 1997).

Faults. Fabrication-related faults can occur from an almost innumerable number of sources. For instance, the etching process relies on the application of liquid photoresist onto each wafer, with subsequent baking to remove solvent and harden the photoresist before it is exposed to UV light through a mask. The UV light must alter the structure of the photoresist, allowing it to be removed by developing. The exposed oxide may then be removed (etched) by techniques such as plasma etching or wet etching. Finally, the dopants are introduced by using an 'ion implanter,' a device that fires dopant ions into the silicon wafer at various depths, depending upon the control setting (measured in keV) (Smith, 1997).

Thus, internal 'open circuit' and 'short circuit' conditions can occur due to such conditions as over-etching and under-etching, respectively. However, they can also occur due to any number of other errors such as misaligned masks, inadvertent use of the wrong mask, chemical contamination, inaccuracies in dopant penetration, etc (Smith, 1997). This plethora of root causes has led to the aforementioned Standards' requirements for 100% 'production-line' testing of safety-critical functions in products that rely upon such devices.

There is also the possibility, however, of very similar failure modes resulting from transient faults, due to capacitive coupling, dynamic cross-linking, electromigration, hot-electron effects, metallization, field oxide parasitism, etc. Detailed examination of these failure mechanisms is beyond the scope of this discussion, but more importantly, we will address the conservative assumption that such failures will occur in a safety-critical system.

USING STANDARDS TO MITIGATE HAZARDS

Scope. The purpose of most Third-Party Safety Standards is to establish a baseline for safety. As such, negotiations are frequently made with the affected industries (particularly during the ANSI canvass process) to define scopes of the Standards. Our example Standard, ANSI/UL 1998 (Software in Programmable Components) has undergone one such major negotiation process resulting in the publication of a second edition.

With the myriad of root-causes, and the seemingly impossible task of establishing test coverage, considerable discussion has taken place, regarding the number of faults that may be considered to generate a given failure mode. Equivalence classes help reduce the amount of testing by utilizing combinatorial logic and network topologies / nodal analysis. Figure 9 provides a simple example.

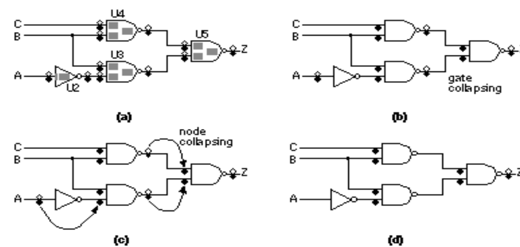

Figure 9. Fault Equivalence (Smith, 1997)

Another practice for reduction of regulatory burden is to categorize the safety-criticality of the Embedded System or partitioned function, resulting in application of either Class 1 or Class 2 software requirements. Appendix A of this Standard details faults for such microelectronic components as registers, program counters, data paths, clocks, memory locations, I/O structures, etc. Not only does this appendix provide guidance regarding faults to be addressed, but it also provides examples of mitigation techniques and their respective applicability. For instance, an appropriately designed pattern test can be used to test not only for 'stuck-at' conditions or 'dc-faults' in memory, but could also be used for verification of the associated data paths.

Thus, using the guidance of such Standards, early in the development process, can not only help assuage regulatory burdens involved with release to market, but can also lead to a more robust, fault-tolerant or fail-safe system.

REFERENCES

International Electrotechnical Commission (IEC), *IEC 60601-1-4 Part 1-4: General requirements for safety – Collateral Standard: Programmable Electrical Medical Systems*, Edition 1.1, IEC, 3, Rue de Varembe, Geneva Switzerland, 2000.

Bezane, Norm, "This Inventive Century," Underwriters Laboratories, Inc., Northbrook Illinois 1994.

Reilly, J. Patrick, *Electrical Stimulation and Electropathology*, Press Syndicate of the University of Cambridge, Cambridge England 1992.

Underwriters Laboratories, Inc. (ULI), "IEC 60601-1-4 Framework for Software Validation in Programmable Electrical Medical Devices." ULI, 333 Pfingsten Road, Northbrook, Illinois, 2000.

Leveson, Nancy, *Safeware- System Safety and Computers*, Addison-Wesley Publishing, Inc., 1995

The Institute of Electrical and Electronics Engineers (IEEE), "Std 1059-1993 IEEE Guide for Software Verification and Validation Plans," IEEE Inc., 345 E 47th St., New York, New York, 1993.

Fusco, John, "Measure Twice, Cut Once," Embedded Systems Programming, CMP, October 2000

Beatty, Sean, "Sensible Software Testing," Embedded Systems Programming, CMP, August 2000.

Ganssle, Jack, G., "As Good As It Gets," Embedded Systems Programming, CMP, January 2002.

Smith, Michael John Sabastian, *Application-Specific Integrated Circuits (ASIC;s…the book)*, Addison-Wesley Publishing Inc., 1997

Suzor, Christophe, "Memory Options," downloaded from semiconductors.net/technical/memories.htm, April 2, 2002.

Underwriters Laboratories, Inc., (ULI), "ANSI/UL 1998 Software In Programmable Components," 2nd Edition, ULI, 333 Pfingsten Road, Northbrook, Illinois, 1999.

BIOGRAPHY

Anura Fernando is a Senior Project Engineer with Underwriters Laboratories, Inc. His current capacity is as the Designated Engineer for ANSI/UL 1998 Software In Programmable Components, at UL's corporate headquarters in Northbrook, Illinois. He has held adjunct faculty positions with both Indiana and Purdue Universities. He holds degrees in Electrical Engineering from Purdue University and Biology/Chemistry from Indiana University. He is currently pursuing a Master's degree in Software Engineering at the University of Maryland.