DOT/FAA/AR-06/34

Office of Aviation Research and
Development
Washington, D.C. 20591

Microprocessor Evaluations for Safety-Critical, Real-Time Applications: Authority for Expenditure No. 43 Phase 1 Report

December 2006

Final Report

This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161.

U.S. Department of Transportation
Federal Aviation Administration

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof. The United States Government does not endorse products or manufacturers. Trade or manufacturer's names appear herein solely because they are considered essential to the objective of this report. This document does not constitute FAA certification policy. Consult your local FAA aircraft certification office as to its use.

This report is available at the Federal Aviation Administration William J. Hughes Technical Center’s Full-Text Technical Reports page: actlibrary.tc.faa.gov in Adobe Acrobat portable document format (PDF).

Technical Report Documentation Page

1. Report No.
DOT/FAA/AR-06/34

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle
MICROPROCESSOR EVALUATIONS FOR SAFETY-CRITICAL, REAL-TIME APPLICATIONS: AUTHORITY FOR EXPENDITURE NO. 43 PHASE 1 REPORT

5. Report Date
December 2006

6. Performing Organization Code

7. Author(s)
Rabi N. Mahapatra and Seraj Ahmad

8. Performing Organization Report No.
TAMU-CS-AVSI-72005

9. Performing Organization Name and Address
Aerospace Vehicle Systems Institute
Texas Engineering Experiment Station
Texas A&M University
Department of Computer Science
College Station, TX 77843-3141

10. Work Unit No. (TRAIS)

11. Contract or Grant No.
DTFACT-03-Y-90018

12. Sponsoring Agency Name and Address
U.S. Department of Transportation
Federal Aviation Administration
Office of Aviation Research and Development
Washington, DC 20591

13. Type of Report and Period Covered
Phase I Final Report
August 2004 – July 2005

14. Sponsoring Agency Code
AIR-120

15. Supplementary Notes
The FAA William J. Hughes Technical Center COTR was Charles Kilgore.

16. Abstract

The intent of this report is to provide findings about safety issues in using today's microprocessors on aircraft. It considers the applicability of RTCA/DO-254 to microprocessors, documents potential safety concerns when using modern microprocessors on aircraft, and proposes potential approaches for addressing these safety concerns.

The project is being performed in two phases with participation from avionic system developers (BAE Systems, The Boeing Company, and Smiths Aerospace) and Federal Aviation Administration organizations responsible for aircraft safety research and development. Phase 1 established the project scope and identified the research parameters as documented in this report.

This report presents an assessment of existing certification guidelines towards certification of microprocessors. It indicates that new validation processes are required in addition to the existing ones. The report identifies that microprocessor obsolescence management may become a significant problem in the future due to rapidly changing designs. This report also addresses unpredictable issues in computational components of the microprocessors that may lead to safety concerns in avionics applications. The microprocessor testing and evaluation trends are presented, and several safety concerns are identified related to the testing and validation.

In the next phase, studies will be made to incorporate a set of recommended guidelines towards selection and qualification of microprocessors in the certification process.

17. Key Words
Microprocessors, Test and validation, Safety, Commercial off-the-shelf, Avionics, Certification

18. Distribution Statement
This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161.

19. Security Classif. (of this report)
Unclassified

20. Security Classif. (of this page)
Unclassified

21. No. of Pages
67

22. Price

ACKNOWLEDGEMENTS

The report authors would like to thank the Aerospace Vehicle Systems Institute (AVSI) Project Management Committee (PMC), for Authority for Expenditure No. 43, and for their participation which contributed to the success of this project. The PMC met regularly, on a bi-weekly basis for more than 2 years, to coordinate the execution of this project. The PMC participation and cooperative effort proved to be productive and beneficial to this project.

The PMC for Phase 1 of this project was as follows:

- Federal Aviation Administration (FAA) contract support: John Lewis, 1st PMC Chair; Robert Manners, 2nd PMC Chair; and Charles Kilgore and Barbara Lingberg, PMC Alternate/Tech Specialists.

- BAE Systems Controls Inc.: Robert Clements and Bob Chobot.

- The Boeing Company: Arnold Nordsieck, Karthikeyan Kesavan, and Dale Snell.

- Smiths Aerospace LLC: Brian Petre, Tom Roe, and Tod Lanham.

- AVSI: Fred Fisher and Reza Langari.

The authors wish to express their appreciation for the assistance of Charles Kilgore of the Federal Aviation Administration who provided valuable comments and suggestions. The authors would also like to acknowledge their deep appreciation for the support of Bob Manners of Apptis who provided significant contributions to this research. Valuable input was also received from participating researchers at Texas A&M University, including Praveen Bhojwani and Ali Chousein.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

LIST OF TABLES

LIST OF ACRONYMS

ABS	Abstract buffer state
APIC	Advanced Programmable Interrupt Controller
ATP	Automated Test Program
ATPG	Automatic test pattern generator or generation
AVSI	Aerospace Vehicle Systems Institute
BCU	Bus Control Unit
BIST	Built-In-Self-Test
BIU	Bus Interface Unit
CAA	Confidence Assurance Architecture
CLA	Check point array
CLB	Configurable logic block
CMOS	Complementary metal-oxide semiconductor
COTS	Commercial Off-The-Shelf
CPU	Central Processing unit
CRC	Cyclic redundancy check
DEU	Dispatch/execution unit
DFT	Design fault taxonomy
DMA	Direct memory access
DRAM	Dynamic random access memory
ECC	Error correction code
ECCU	Error correction code unit
ED	Error detection
ELR	Error logging/reporting
ER	Error recovery
FAA	Federal Aviation Administration
FD	Fault diagnosis
FDU	Fetch/decode unit
FI	Fault injection
FPGA	Field programmable gate array
I/O	Input/output
IP	Intellectual property
IR	I-Current times R-resistance
L1C	L1 Data and Instruction Cache
L2C	L2 Cache
LRU	Line replaceable unit
MA	Millicode Array
MMU	Memory management units
MP	Multiprocessor
OA	Other Arrays
OEM	Original equipment manufacturer
PDA	Performance Delivery Architecture
ROM	Read-only memory
RU	Retire Unit

SB	Store Buffer
SoC	System-on-a-chip
UP	Uni-processor
VLSI	Very large-scale integration
WCET	Worst-Case Execution Time

EXECUTIVE SUMMARY

Progressively complex microprocessors originally developed for consumer, automotive, and industrial uses are being used in aviation applications. These microprocessor devices reduce the size, weight, and power requirements of a product and add capability by using advanced design and dense component packaging techniques. However, evolving microprocessor architectures include concepts, such as caching and pipelining, which can affect system predictability and safety. This is especially true as more complex microprocessors are being used, more complex hardware is integrated, and fully partitioned systems are being implemented. Thus a defined process for microprocessor evaluation and acceptance is needed.

This research focuses on current microprocessors being proposed on aircraft and establishes evaluation criteria. The project (1) considers the applicability of RTCA/ DO-254 to microprocessors, documents potential safety concerns when using modern microprocessors on aircraft, and proposes potential approaches for addressing these safety concerns; (2) considers issues of modern microprocessor architecture and related system architectures and their use in integrated modular avionics; and (3) provides practical techniques for use by aircraft manufacturers, avionics developers, certification authorities, and other stakeholders. The project also provides criteria for the level of rigor required for various levels of systems that use microprocessors (i.e., to provide an approach for scaling the criteria depending on the functional criticality of the processor). The results will be used by the Federal Aviation Administration (FAA) to develop policy, regulations (if deemed needed), and guidance materials for industry. The output may be expanded upon in the future to address other commercially available complex devices.

This project is being performed in two phases by a combination of avionics system developers (BAE Systems, The Boeing Company, and Smiths Aerospace) and FAA organizations responsible for aircraft safety research and development.

Phase 1 established the project scope and identified the research parameters as documented in this report: (1) survey of industry and government organizations responsible for the development and certification of avionics systems, (2) search of relevant literature, (3) detailed evaluation of existing FAA policy and guidance, and (4) development of subject white papers that provides the basis for Phase 2, as shown in the list below.

- Safety Issues and mitigation strategies related to modern microprocessor architectures
- RTCA/DO-254 suitability for the evaluation of microprocessors
- Issues unique to commercial-off-the-shelf microprocessors
- Evaluation criteria for microprocessors
- Issues related to microprocessor obsolescence

- Evaluation criteria based on criticality levels

- Comprehensive survey of strategies for worst case execution time in the presence of unpredictable computation components such as pipeline, caches, and branch prediction.

- Approaches and strategies for microprocessor test and validation

- Issues related to System-on-a-Chip architectures

Candidate microprocessors have been selected for detailed case studies in Phase 2 and will provide a detailed analysis of these candidate microprocessors to establish and document approaches for evaluating microprocessors to ensure that the safety issues have been addressed. Phase 2 will also provide evaluation criteria specific to microprocessors that may be used to comply with RTCA/DO-254 or to serve as input to an FAA Advisory Circular or as an update to RTCA/DO-254. These evaluations will be based on test scenarios and test cases to be applied to test environments for the candidate microprocessors.

Evaluation criteria and methods for microprocessors in avionics systems and applications will be developed, researched, and documented, including refinement of deterministic methods and evaluation of performance and functionality. This report will be updated and the results of Phase 2 will be refined into a Microprocessor Evaluation Handbook to provide input to the FAA policy and guidelines for evolving modern microprocessors.

Management controls have been established to permit the FAA to publish safety issues and guidance while protecting the rights of all partners to the project technology developed during this undertaking.

1. INTRODUCTION.

The Aerospace Vehicle Systems Institute (AVSI) and Federal Aviation Administration (FAA) as partners have sponsored this research that focus on evaluation techniques to address the safety concerns when using Commercial Off-The-Self (COTS) microprocessors in avionics. BAE Systems, The Boeing Company, and Smiths Aerospace, industry partners of AVSI, have actively participated in this effort. Industry goals included effective ways to select, configure, and apply current and future microprocessors to cost-effective and safety-critical avionics applications. FAA goals included preparing policy and guidance for effective safety certification and use of newer microprocessors in future avionics systems.

This project evaluates the use of modern microprocessors in airborne systems in two consecutive phases. The first phase identifies the features and issues related to safety in the aeronautical use of microprocessors. The second phase will establish and evaluate the methods to mitigate risk and ensure safety in the certification of these systems and products on aircraft.

The safety issues of using microprocessors in avionics are outlined in section 2. The RTCA/DO-254, Design Assurance Guidance for Airborne Electronic Hardware [1], provides guidance that ensures that airborne electronic hardware performs its intended functions in the specified environment. From this point on RTCA/DO-254 will be referred to as DO-254. Section 3 of this report examines the suitability of that document for evaluating the use of microprocessors in aircraft. The general conclusion is that DO-254 contains guidance for the evaluation of COTS microprocessors that will probably not be adequate to evaluate future microprocessor technologies and architectures. These inadequacies are beginning to be experienced now and are reflected in the need to evaluate performance and functionality in cases where time-dependent predictability becomes obscure. Future evaluation criteria are proposed in section 4.

Processor evaluations should be done in a given application context, which requires the scaling of evaluation criteria for various levels of applications categorized according to safety-criticality. Section 5 classifies the evaluation criteria in safety levels A to E.

Electronic component obsolescence is a major issue in the avionics industry. However, it encompasses all variety of electronic circuits. Section 6 discusses microprocessor obsolescence and presents a survey of available approaches and discusses their advantages and drawbacks.

Section 7 compiles existing modeling approaches of advanced features of microprocessors for making worst-case execution time (WCET) predictions.

Microprocessor testing and validation issues are presented in section 8. A study into the system-on-a-chip (SoC) integration issues for microprocessors in SoCs is presented in section 9. The conclusion of the study carried out by AVSI and the FAA is presented in section 10.

2. SAFETY ISSUES WHEN USING MICROPROCESSORS IN AVIONICS.

2.1 OVERVIEW.

This section discusses the safety issues when using microprocessors in avionics products. There are several issues that are classified under different categories. The following sections consider each category and further elaborate on the related safety issues.

2.2 AVAILABILITY AND VALIDITY OF INFORMATION.

Information about microprocessors is usually available in the form of user or reference manuals. These documents are provided by manufacturers to the customers to give them information about the usage of the microprocessor. Information about the design, production, testing, and validation accomplished by the manufacturer are not included because it is considered proprietary information by the manufacturer/vendor. Thus, studying the documentation alone is not sufficient to conclude that the design, production, and testing were all rigorous enough to guarantee that the microprocessor will function correctly in harsh or critical environments. Additionally, there may be some logic implemented on the microprocessor that has not been documented because it is used by the manufacturer for in-house testing. These undocumented features may cause unpredictable execution. Hence, the amount of information made available by the manufacturer is an issue when deciding which microprocessor to use in an avionics product.

Validity of the information provided by the manufacturer also causes some concerns. Different types of invalidity may arise. The documentation may give incorrect information about a feature and this fact may not appear in the periodically released errata documents. Documenting a nonexistent feature is another type of invalidity. A manufacturer may decide not to implement a feature that initially was planned for, and if the information about this feature is not excluded from the documentation, the contents of the manuals will be misleading. Such types of incorrect information are undesirable when a microprocessor is used in critical applications such as avionics.

2.3 MANUFACTURER TRUSTWORTHINESS.

Trustworthiness of the manufacturer refers to the company's life expectancy, its financial stability, and its experience in working with and cooperating with military and avionics industries. Using products of a manufacturer that goes out of business causes serious risks for an avionics product that on average has had a 30-year life expectancy. Lack of experience in a similar application type is another concern because issues left unresolved by an inexperienced manufacturer may be repeated.

2.4 MICROPROCESSOR TRUSTWORTHINESS.

Trustworthiness of a microprocessor is decided by the different properties that it has or lacks. Predictable execution time of programs running on a microprocessor and their obsolescence are important topics that are discussed separately in sections 3 and 4. Expected lifetime of a microprocessor predicts the time that it will become obsolete. The period of time it has been in the market is another trustworthiness criterion. The microprocessors that have been in the market longer tend to be more robust because they have been extensively tested by being used in various applications fields. This is not the case for newly released microprocessors because they have been tested by the manufacturer only. If a manufacturer has not considered testing certain environmental scenarios, then the microprocessors may be unsuitable for avionics products. Testing the microprocessor in temperature ranges that arise in avionics environments is an example of such a situation. The amount of support given by a manufacturer and the cost of this support is also important. Assuring aircraft safety requires the number of faults be minimal. However, ensuring minimum number of faults may require the support of the manufacturer because the manufacturer possesses all the information about the design, production, and testing of the microprocessor. Prompt notification of errata is also important.

The components such as branch prediction, caches, pipelining, out-of-order execution, interrupt and interrupt masking, error detection and correction, and parity protection of different components (busses) are discussed in sections 2.5 and 2.6.

2.5 UNPREDICTABLE EXECUTION AND WCET.

The execution behavior of a microprocessor can be unpredictable due to some of the advanced features it incorporates. This section compiles the list of the advanced features that are known to cause unpredictable execution times.

Cache memories (Instruction, Data, L2 Cache (L2), Translation Look-Side Buffer, etc.) make the WCET of the running tasks difficult to predict because of the intertask and intratask interference caused through them. In a multitasking environment, cache memories are shared resources and a cache line may be used by several tasks. Intertask interference can occur during a context switch, where a newly scheduled task changes the cache contents by replacing existing entries used by other tasks. However, this interference uncontrollably affects the execution times of tasks due to unpredictable capacity cache misses. Intratask interference is caused because of capacity and conflict cache misses. It is hard to predict the WCET of tasks unless the ratio of these misses is modeled and tightly bounded. Cache memories cause unpredictable execution because intertask interference violates the principle of address partitioning. Address partitioning is used for achieving security, and it requires that tasks do not affect the execution of each other uncontrollably. But when the cache is a common resource used by all tasks without any restriction, the principle of address partitioning might not be achieved.

Pipelining of instruction and branch prediction are among the advanced features that make WCET of tasks hard to predict. Incorrect branch predictions stall the pipeline and this increases the execution times of tasks. If the number of incorrect branch predictions is not modeled and tightly bounded from above, the WCET time of a task cannot be tightly bounded either.

Out-of-order instruction execution or dynamic scheduling of instructions may cause timing anomalies. For instance, when there is a cache hit, an instruction takes longer to execute than when there is a cache miss, contrary to popular knowledge that cache hits take less time. For example, in a processor that employs out-of-order execution, a cache miss will allow subsequent instructions to begin execution. This out-of-order behavior may lead to a reduced execution time for a set of instructions. This makes the WCET of tasks hard to predict. Reference 2 gives an example of this anomaly.

Interrupts are another source for WCET unpredictability. Occurrence of an interrupt causes a context switch and hence intertask interference. If enabled, an interrupt can occur at any time during normal execution and trigger a context switch leading to unpredictability.

Other sources of WCET uncertainty are multimaster/arbitration for external busses and simultaneous use of multiple Direct Memory Access (DMA) engines. The former can result in arbitration issues that can delay the completion of transactions while the latter can result in interleaving and nondeterministic completion of DMA transactions.

2.6 FAULT TOLERANCE SUPPORT.

Fault tolerance support is important to ensure that the microprocessor continues to function correctly even when faults occur. Several features of a microprocessor need to have fault tolerance support. Internal address Data Tag, Register parity, and/or Error Correction Code (ECC) protection, and external interfaces (busses) supporting parity protection on address and data are crucial for fault tolerance.

2.7 MICROPROCESSOR OBSOLESCENCE.

Microprocessors are among the most popular packaged electronic components and observe fierce market competition. Leading manufacturers always try to stay ahead of the competition by frequently introducing higher-performance products. Microprocessors have several performance, design, and process attributes that have different values in different versions or stepping (minor revisions) of a microprocessor. The safety-critical characteristics of the avionics applications are highly sensitive to the value of these attributes, which makes the microprocessor obsolescence much more likely for avionics systems. The short product life of microprocessors is a concern in the avionics and other safety-critical systems that tend to have a very long projected operational life.

Thus, microprocessor obsolescence must be treated as an integral part of the avionics product lifecycle. The strategies dealing with microprocessor obsolescence attempt to manage the problem at several levels starting with techniques for predicting the microprocessor obsolescence, pushing the actual event of obsolescence by performing a life-time buy of the microprocessor, and various system modifications to accommodate a newer microprocessor. Section 6 deals with the approaches to mitigate microprocessor obsolescence and associated safety issues.

2.8 SAFETY ISSUES.

The following list summarizes the safety issues raised when using a microprocessor in avionics products.

- Available information about a microprocessor might not give sufficient information on all of its features.

- Available information about a microprocessor might be invalid.

- The manufacturers of a microprocessor might not be trustworthy because of their:

  - History
  - Expected lifetime
  - Financial stability
  - Lack of experience in similar applications

- A microprocessor might not be trustworthy because of its:

  - History
  - Expected lifetime
  - Nonuse in similar applications
  - Support given by the manufacturer
  - Late notification of errata lists
  - Advanced features that cause unpredictability in execution times of tasks
  - Fault tolerance support

- Runtimes of tasks running on a microprocessor might not be predictable because of:

  - Cache memories
  - Branch prediction
  - Out-of-order instruction execution
  - Interrupts
  - Multimaster/arbitration for external buses
  - Simultaneous use of multiple DMA engines

- The fault tolerance support of a microprocessor might not be sufficient

- Microprocessors become obsolete much before the lifetime of an avionics product ends

- Failure of highly integrated microprocessors including input/output (I/O) controllers (single point failures with potentially significant impact)

- Limited Built-In, Self-Test (BIST) support

- Limited performance and functionality testing
- Lack of mechanism to handle microprocessor warning, failure, and error messages
- Confidence in a microprocessor obtained from a primary manufacturer may not be extended to the same microprocessor obtained from secondary manufacturing sources
- Lack of adequate testing of high-performance microprocessor containing many advanced features due to exploding testing complexity
- Submicron proximity related problems in highly integrated microprocessors

3. DO-254 SUITABILITY FOR MICROPROCESSORS.

3.1 OVERVIEW.

This section comments on the suitability of DO-254 for evaluating microprocessors. DO-254 provides guidance that ensures that airborne electronic hardware safely performs its intended functions in the specified environment. DO-254 considers all stages that are necessary to develop a hardware product. The spectrum of the stages ranges from the hardware planning processes, to the hardware design and implementation processes, to the hardware validation and verification processes, and finally to the hardware maintenance processes. This wide spectrum of guidance is not applicable for assessing the suitability of microprocessors in critical environments, because the manufacturer of the microprocessors may not follow the guidance give in DO-254. If microprocessor vendors do follow DO-254 guidance, it has been the general practice for them not to release detailed information to protect the vendor's proprietary information. Additionally, DO-254 does not consider some safety issues discussed in section 2 of this document. Hence, DO-254 cannot normally be used as a basis for accepting or rejecting the usage of a given microprocessor in the avionics domain. However, the objectives of DO-254 may still be used as optional guidance, that if followed, gives additional confidence about the credibility of a given microprocessor.

The analysis of DO-254 suitability for evaluating microprocessors is presented in sections 3.2, 3.3, and 3.4. Section 3.2 discusses the sections of DO-254 that are suitable for microprocessors. Section 3.3 comments on sections of DO-254 that are not suitable for microprocessors. The main reason of the unsuitability is the lack of information required by the corresponding sections of DO-254. Section 3.4 lists the safety issues for which DO-254 does not give guidance for mitigating them.

3.2 SUITABLE SECTIONS OF DO-254 FOR EVALUATING MICROPROCESSORS.

The guidance given in section 2, System Aspects of Hardware Design Assurance, of DO-254 is suitable for evaluating microprocessors. The hardware related information required by the guidance on the "Information Flow From System Development Process to Hardware Design Life Cycle Process," "Information Flow From Hardware Design Life Cycle Process to System Development Process" and "Information Flow Between Hardware Design Life Cycle Process and Software Life Cycle Process" can be obtained through sources discussed in section 4.2 of this report. For example, probabilities for hardware functional failures and failure conditions for each function can be obtained by the user by directly testing and verifying the microprocessor. Hardware safety assessment, quantitative assessment of random faults, and design errors and anomalies are possible by directly testing and verifying the hardware in the user environment. For long-term failures due to aging and environmental stress, the accelerated testing approaches such as burn-in and environmental stress testing (as explained in section 8 of this document) can be applied.

Section 6, Validation and Verification Process, of DO-254 describes the validation and verification process. Two types of validation and verification processes may be considered for a microprocessor: validation and verification done by the manufacturer and validation and verification done by the applicant.

When the validation and verification done by the manufacturer is considered, section 6 of DO-254 is not applicable for evaluating microprocessors. Information about the validation and verification done by the manufacturer is confidential and usually not accessible. Even if this information is accessible, the validation and verification done by the manufacturer might not necessarily meet the guidance of DO-254. However, this does not necessarily mean that the quality of a COTS microprocessor is not suitable for avionics. For alternative criteria for evaluating microprocessors, refer to section 4 of this report. If the validation and verification process information of the manufacturer is accessible, in addition to the alternative criteria in section 4 of this document, the objectives in section 6 of DO-254 may be used as optional criteria.

Validation and verification of a microprocessor can also be done by the applicant. The applicant should consider the evaluation criteria specified in section 4 of this report and make sure that the available sources of information are valid, the manufacturer and the microprocessor are trustworthy, the features that support predictable execution and fault tolerance function correctly, and the workarounds to design faults produce the expected execution.

Section 7, Configuration Management Process, of DO-254 is applicable for evaluating microprocessors as there are many artifacts such as test patterns, performance profile and diagnostic programs, and other associated data that need to be managed similar to configuration items. This step is important in implementing a successful microprocessor obsolescence management strategy as well.

Section 11, Additional Considerations, of DO-254 is applicable for microprocessors. The guidance on use of previously developed hardware should be used whenever a new stepping (corresponds to a new mask version which has been tuned according to more matured process parameters and incorporates minor fixes for field-reported problems) of a microprocessor is released and will be used in an avionics product. Section 11 of DO-254 has dedicated subsections that give guidance on microprocessor components usage and product service experience. These subsections are directly applicable to COTS microprocessors but do not give guidance on mitigating all the safety issues. The guidance given in DO-254 on tool assessment and qualification is not applicable to microprocessors when the tools used by the manufacturer are considered. However, when the tools used by the user for testing and verification of the microprocessor are considered, then guidance of DO-254 becomes applicable.

3.3 UNSUITABLE SECTIONS OF DO-254 FOR EVALUATING MICROPROCESSORS.

Section 4, Planning Process, of DO-254 is not applicable for evaluating microprocessors. Information about the hardware planning process for controlling the development of microprocessors probably will be lacking because the manufacturer will not be willing to share this information. Even if the manufacturer shares this information, the hardware planning process of the manufacturer may not overlap with the guidance of DO-254. However, the hardware planning process of the manufacturer might still be good enough such that the final product (the COTS microprocessor) meets the requirements of high-criticality levels. Similarly, section 5, Hardware Design Process, of DO-254 is not applicable for evaluating microprocessors. Information about the hardware design process of a microprocessor will be lacking because it is confidential to the manufacturer. It is possible to develop a model for studying the reliability of the design of a microprocessor (see section 5.7 of this report).

Section 8, Process Assurance, of DO-254 is not applicable for COTS microprocessors. Information about the life cycle process of a microprocessor will be lacking because it is confidential to the manufacturer. Besides, the general market mainly determines the life cycle of microprocessors and the microprocessor demand of the avionics industry is negligible compared to the demand of the general market (e.g., demand of the cell phone industry).

Section 9, Certification Liaison Process, of DO-254 is not applicable for microprocessors because the certification authority will not be able to provide input to the hardware design life cycle. A manufacturer of microprocessors considers the general market demand and will not establish a communication with the certification authority because that will require him to concentrate only on a small portion of the market.

3.4 SAFETY ISSUES NOT CONSIDERED BY DO-254.

DO-254 does not give guidance on the following safety issues:

- Certifying the availability and validity of information on a microprocessor
- Assessing the trustworthiness of a manufacturer
- Assessing the trustworthiness of the microprocessor
- Mitigating WCET unpredictability caused by advanced features
- Assessing the degree of fault tolerance support
- Mitigating microprocessor obsolescence
- Use of performance and functionality testing in lieu of predictability (WCET)

4. EVALUATION CRITERIA FOR MICROPROCESSORS.

4.1 OVERVIEW.

This section discusses the sufficient criteria for evaluating the suitability of a microprocessor in avionics. The evaluation criteria specified in this section should provide a basis for the development of standards, guidelines, and processes to be used in lieu of the guidance of DO-254. The evaluation criteria are divided in the following categories:

- Availability and validity of information
- Trustworthiness of a manufacturer
- Trustworthiness of a microprocessor
- Predictable execution
- Fault tolerance support
- Reliability of the design
- Availability and suitability of tools
- Adequate mitigation for microprocessor obsolescence
- Performance and functional testing

The following sections discuss each category.

4.2 AVAILABILITY AND VALIDITY OF INFORMATION.

Availability of information about a microprocessor is crucial for understanding its features. Information can be obtained from different sources:

- Public documentation provided by the manufacturer

- Published case studies

- Information obtained from the manufacturer under nondisclosure agreement

- Direct testing and evaluation of the microprocessor in test bed and/or in the target application

The criteria that can be used for validating the available information are discussed below. Possible ways of validating public documentation provided by the manufacturer are the following:

- Compare the release date of the public documentation against the release date of the corresponding microprocessor. If the release date of the public documentation is before the release date of the microprocessor (6-12 months), the documentation could be outdated.

- Contact the manufacturer and ask if the available public documentation is outdated. The customer support services should be able to answer this question.

- Compare the public documentation against the published case studies. This will show the inconsistencies (if any) between them. Any such inconsistency is an indication that the public documentation includes invalid information.

The criterion for evaluating published case studies is that the reported results should be reproducible. Enough information should be given in the case studies so that the user can obtain the reported results.

Information obtained from the manufacturer under nondisclosure agreement can be assumed to be reliable in the sense that it is less likely to be misleading because of outdated or incomplete information. The main issue with this type of information is that the manufacturer may request payment in exchange for this information. The cost of buying (if applicable) confidential information about a microprocessor could be prohibitive. Some manufacturers might not agree to sell any sensitive information about their products even for a high cost. The risks of a manufacturer's refusal to sell the information and the cost of keeping the information confidential should also be considered.

4.3 MANUFACTURER TRUSTWORTHINESS.

Business research should be done to answer the following questions about the trustworthiness of a manufacturer (ordering does not imply priority):

- How long has the manufacturer been in business?

- How long the manufacturer is expected to remain in business?

- Is the manufacturer International Organization for Standardization 9001 certified?

- Is the manufacturer financially stable?

- Does the manufacturer currently support or has previously supported a full Military Temperature/Defense Supply Center Columbus product line (extended temperature ranges, industrial/automotive/full military)?

- Previous design history/experience with similar product lines/environments.

Manufacturers that satisfy the above criteria should be given priority over others, everything else being equal.

4.4 MICROPROCESSOR TRUSTWORTHINESS.

Research should be done to answer questions about the trustworthiness of a microprocessor (ordering does not imply priority). Some listed items are further elaborated in the reference sections.

- How long the microprocessor has been in the market?
- What is the expected lifetime of the microprocessor?
- Is the microprocessor used in a similar application type?
- What is the temperature range in which the microprocessor can operate?
- Previous up screening experience and yield information (i.e., up screen industrial temperature part to military temperature range).
- Does the manufacturer provide support? What kind of support is available? Are problem histories made available?
- What point will the support be withdrawn or incur higher cost?
- Is there a warranty, and what limits exist on this warranty?
- Is there prompt and automated notification support of updated design, process, die, and errata information?
- Are memory management units (MMU), hardware, and software partitioning supported?
- Are nonmaskable interrupt(s) supported?
- Does the microprocessor have components (e.g., cache, branch predictor) that would cause the execution time to be unpredictable? (See section 4.5)
- Is there internal address, data, tag, and register parity, or ECC protection? (see section 4.6.1.)
- Are there internal or external memory controller/bridge controller support (required companion chips)?
- Do external interfaces (busses) support parity protection on address and data?

4.5 PREDICTABLE EXECUTION.

Assessing predictable execution of a microprocessor requires identifying both its advanced features that cause unpredictability and the attributes that it has for supporting remedies. If there is a remedy for each type of unpredictability, then the microprocessor can be configured to execute predictably. The possible remedies to causes of unpredictability discussed in section 2.5 are discussed below.

WCET can be unpredictable either because of intertask interference or intratask interference through the cache. Possible remedies are:

- If the microprocessor supports turning off the cache, the cache may be turned off. This approach solves the address space partitioning violation problem as well. However, this is undesirable because it degrades the system performance considerably.

- If the microprocessor supports the cache-locking feature, cache locking can be used for eliminating the effects both due to intertask and intratask interferences.

- Partitioning the cache among different tasks is a method for eliminating the intertask interference effects. Current microprocessors, (as of September 2005) do not provide hardware support for partitioning the cache. An alternative is to use software-based cache partitioning, as explained in reference 3. However, this needs compiler support, as explained in reference 4. Cache partitioning solves the address space partitioning violation problem as well.

- If the microprocessor supports flushing and invalidation of cache lines, the cache can be emptied upon a context switch. This eliminates the effects due to intertask interference because each time a new task is scheduled the cache is empty. It also solves the address space partitioning violation problem.

- Model the cache (as explained in section 7.2 of this report) to make WCET prediction more precise.

The associativity of the cache may affect the applicability of some of the remedies. For example, software cache partitioning is not suitable for high-associativity caches because software-based cache partitioning works best with direct-mapped caches.

Possible remedies to uncertainty caused by incorrect branch predictions are:

- If the microprocessor supports turning the branch prediction off, turn it off. This will cause pipeline stalls for each fetched branch instruction and will reduce the parallelisms of the code. However, the unpredictability caused by the branch instruction is eliminated.

- Model the branch prediction (as explained in section 7.3 of this document) to estimate the upper bound of the number of incorrect branch predictions. This will also give an upper bound for the number of pipeline stalls due to incorrect branch predictions. WCET estimation uses this bound for making timing predictions. The number of pipeline stalls can be predicted by modeling the pipeline as well. This subject is discussed in section 7.4 of this report.

Enforcing in-order execution can eliminate the uncertainties caused by out-of-order instruction. This usually requires hardware support in the microprocessor in the form of an in-order instruction issue logic core in addition to the out-of-order issue logic core. Another technique might be to introduce pseudodependency in the instruction to prevent it from being scheduled out of order. However, this will require in-depth, control-flow, and data-flow analyses of the source program and internal knowledge about the out-of-order issue logic and may eventually turn out to be a costlier solution due to decreased application performance introduced by pseudodependencies.

4.6 FAULT TOLERANCE SUPPORT.

4.6.1 Overview.

The analysis discussed here adopts the techniques developed in references 5 and 6 for assessing the applicability of microprocessors in high-confidence systems. According to references 5 and 6, a COTS microprocessor may be applicable in high-confidence systems if it supports fault tolerance (error detection and recovery) features such that it is assured that it continues to function correctly in the presence of faults. In references 5 and 6, information is collected from publicly available technical documentation only; however, each type of information specified in section 4.2 of this report can be used. The techniques answer the following questions:

- Which built-in fault tolerance features are available in a microprocessor (i.e., internal and external bus address/data parity or ECC coverage, internal register parity, internal and external cache address/data/tag parity or ECC coverage, hardware or software watchdog monitor, and redundant clock inputs)?

- How effective are the built-in fault tolerance features?

- What mechanisms are available to test these built-in, fault-tolerant features?

The approach consists of three modeling elements. Each modeling element is discussed in the following sections. The case studies considered here as examples study the Intel Pentium® II and IBM S/390 microprocessors. Similar studies will be conducted for the MPC8540 and PowerPC 7447 microprocessors chosen as case studies for Phase 2.

4.6.2 Fault Model.

The fault model gives the list of faults to be evaluated for acceptable levels of risk. The case study in reference 5 gives two examples. For the Intel Pentium® II processor, the fault model list contains:

- Recoverable errors
- Unrecoverable errors
- Fatal errors

For the IBM S/390 G5, the fault model is:

- Permanent errors
- Transient faults

4.6.3 System Model.

The system model consists of four parts:

- Performance delivery architecture (PDA)
- Confidence assurance architecture (CAA)
- Operation modes
- Configuration

The PDA is the logic part of the processor that is responsible for delivering the documented performance. Following are the PDAs for Intel Pentium II and IBM S/390 respectively as presented in reference 5.

- Intel Pentium II PDAs:

    - Fetch/Decode Unit (FDU)
    - Dispatch/Execution Unit (DEU)
    - Retire Unit (RU)
    - L1 Data & Instruction Cache (L1C)
    - L2 Cache (L2C)
    - Bus Interface Unit (BIU)
    - Advanced Programmable Interrupt Controller (APIC)

- IBM S/390 PDAs:

    - I-Unit
    - E-Unit: Fixed-Point Unit and Floating-Point Unit
    - Buffer Control Unit (BCU): store buffer (SB) and other arrays (OA)
    - BIU

The CAA is the logic part of the processor that is responsible for assuring that the PDA delivers the documented performance in the presence of faults. The functions that it provides are:

- Error detection (ED)
- Error recovery (ER)
- Error logging/reporting (ELR)
- Fault diagnosis (FD)

Tables 1 and 2 give the CAAs for Intel Pentium II and IBM S/390 respectively as presented in reference 5.

Table 1. Intel Pentium II CAAs

CAA Functional Unit	Functions
Voltage Monitoring Unit	ED
Thermal Monitoring Unit	ED
Parity Checking Unit	ED
Functional Redundancy Checking Unit	ED
Error Correction Code Unit	ED, ER
Watchdog Timer	ED
Checksum Unit	ED
Memory Protection Unit	ED
Reasonableness Checking Unit	ED, ER
Retry Logic	ER
Machine Check Architecture	ED, ER, ELR
Built–In, Self–Test Unit	FD
Test Access Port	FD
Debugging Unit	FD

Table 2. IBM S/390 CAAS

CAA Functional Unit		Functions
Parity Checking Unit		ED
Error Correction Code Unit		ED, ER
Checksum Unit		ED
Memory Protection Unit		ED
Reasonableness Checking Unit		ED
Timer		ED
R-Unit	Comparison Logic	ED
	Checkpoint Array	ER
	Retry Logic	ER
	Trace Array	FD
Memory Scrubbing Logic		ER
Cache Line Delete/Relocate Logic		ER
Machine Check Logic		ED, ELR
Built–In, Self–Test Unit		FD

The examples given in reference 5 indicate that Pentium II processor has five and IBM S/390 G5 has six operation modes. The following are the operation modes for Pentium II:

- Start
- Test
- Normal
- System management mode
- Recovery

The following are the operation modes for the IBM S/390 G5:

- Load
- Test
- Operating
- Recovery
- Stop
- Check-stop

The case study in reference 5 specifies that the Pentium II processor can be configured in three different ways: Uniprocessor (UP), Multiprocessor (MP), and Functional redundancy checking.

IBM S/390 G5 has two main configurations: UP and MP.

Both configurations of IBM S/390 G5 have three subconfigurations: UP/MP with spare only, UN/MP with service assist processor only, and UP/MP with both spare and service assist processor.

4.6.4 Effectiveness Measure.

The fault model, the system model, and the public documentation are used for building two coverage matrices: PDA Coverage Matrix and the CAA Coverage Matrix.

The rows of the PDA coverage matrix consist of PDA elements identified in the system model. The columns consist of the confidence assurance functions (ED, ER, ELR, and FD). Similarly, each row of the CAA coverage matrix corresponds to one CAA element identified in the system model, and the columns consist of the confidence assurance functions. The effectiveness measure is conducted by deriving the entries of the PDA and CAA coverage matrices.

The PDA coverage matrices for the Pentium II and IBM S/390 G5 were developed in reference 5 and are given in tables 3 and 4. According to reference 5, the Pentium II CAA is not covered by the confidence assurance functions and only the check-pointing array of the IBM S/390 G5 is covered. Hence, the CAA coverage matrices for both processors are not given.

Table 3. Performance Delivery Architecture Coverage Matrix for Intel Pentium II

	Confidence Assurance Functions		
PDA Element	ED	ER	ELR
FDU	PCU	NC	MCA
DEU	NC	NC	NC
RU	WDT	Reset	MCA
L1C	PCU	NC	MCA
L2C	PCU	NC	MCA
BIU	PCU	NC	MCA
	ECCU	ECCU	MCA
	Protocol	RL	MCA
APIC	CSU	RL	Y

NC = Not Covered
PCU = Parity Checking Unit
ECCU = Error Correction Code Unit
WDT = Watchdog timer
CSU = Check sum unit
MCA = Micro Channel Architecture
RL = Retry Logic

Table 4. Performance Delivery Architecture Coverage Matrix for IBM S/390

		Confidence Assurance Functions			
			ER		
PDA Element		ED	TF	PF	ELR
I–Unit		dup./comp.	RL	sparing	MCL
E–Unit		dup./comp.	RL	sparing	MCL
BCU	MA	PCU	RL	sparing	MCL
	SB	ECCU	ECCU, RL	sparing	MCL
	OA	CRC	RL	sparing	MCL
BIU		ECCU	ECCU, RL	sparing	MCL

MA = Millicode Array
RL = Retry Logic
MCL = Machine Check Logic
TF = Temporary Fault
PCU = Parity Checking Unit
ECCU = Error Correction Code Unit
PF = Permanent Fault

4.7 RELIABILITY OF THE DESIGN.

The analysis discussed in this section adopts the design fault taxonomy (DFT) technique introduced in reference 6 for analyzing the effects of the documented design faults on the suitability of a microprocessor in critical applications. The DFT proposed in reference 6 consists of the following four actions.

a. Identification of design fault: This initial step consists of collecting all known design faults, usually published as an errata list by the manufacturer.

b. Development of target system model: The PDA and the CAA discussed in section 4.6 of this report constitute the target system model. These modeling models are at the highest abstraction level. If information is available and there is need for more detailed modeling, the PCA and CAA models can be elaborated further.

c. Development of design fault model: A design fault is characterized based on the following attributes:

    (1) Logical location of the design fault: Design faults can be found in the PDA or the CAA.

    (2) Type of the design fault: Describes the type of the error that is caused by the design fault. Possible types of errors are timing, data, and control.

    (3) Triggering condition of the design fault: Describes the environment under which a design fault affects the correct execution of the microprocessor. It is divided into three categories:

        (a) Configuration: Describes the system configuration when a design fault affects correct execution. Uniprocessor or multiprocessor configurations are two examples.

        (b) Operation mode: The operation when a design fault affects correct execution. Examples are normal, test, and recovery.

        (c) Triggering dependency: Describes the preconditions for a design fault to affect correct execution.

    (4) Effect of the design fault: This is the list of errors that are caused by a design fault. The design fault is further divided into two categories:

        (a) Severity: Describes the severity of an error caused by a design fault.

        (b) Affected elements: Describes the list of logic elements, for example function(s) and instruction(s), that are affected by a design fault.

d. Classification procedure: After the above three steps are completed, each fault of the design fault list is analyzed using the design fault model.

4.8 AVAILABILITY AND SUITABILITY OF TOOLS.

Microprocessors that have the following tools available should be given priority over others that do not have such tools (ordering does not imply priority order):

- Compilers and linkers
- Debuggers
- Performance analysis tools
- JTAG/In-Circuit Emulators

However, the evaluation of the suitability of these software tools for avionics is beyond the scope of the Phase 1 study.

5. EVALUATION CRITERIA BASED ON CRITICALITY LEVELS.

5.1 OVERVIEW.

This section classifies the evaluation criteria of section 4 in safety Levels A to E. DO-254 defines the software criticality levels based on the severity of failures that will be caused by an anomalous behavior. The following list summarizes the criticality levels:

- Level A: Anomalous behavior that would cause a catastrophic failure condition for the aircraft. A failure is catastrophic when it prevents continued safe flight or landing.

- Level B: Anomalous behavior that would cause a hazardous/severe-major failure for the aircraft. A failure is hazardous/severe-major when it reduces the capability of the aircraft or the ability of the crew to manage adverse operational conditions to the extent that there would be a large reduction in safety margins or higher workload that would affect the capability of the flight crew to perform their tasks accurately and completely or adverse effects on occupants that include fatal injuries.

- Level C: Anomalous behavior that would cause a major failure for the aircraft. A failure is major when it reduces the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to such an extent that there is a significant reduction in safety margins or functional capabilities or a significant increase in crew workload or discomfort to the occupants to the extent that it would cause injuries.

- Level D: Anomalous behavior that would cause a minor failure for the aircraft. A failure is minor when it does not reduce the aircraft safety significantly and involves crew actions that are well within the crew's capabilities.

- Level E: Anomalous behavior that would have no effect on the operational capability of the aircraft or crew workload.

5.2 AVAILABILITY AND VALIDITY OF INFORMATION.

Availability of information for a microprocessor is very significant for understanding the features it has. Lack of comprehensive knowledge of a microprocessor that is used in an avionics system may result in a catastrophic or a hazardous/severe-major or a major failure. Hence, for high-criticality levels (Levels A, B, and C), the following valid information should be available for selected microprocessors.

- All the features of the microprocessor

- The verification of the computational correctness of the microprocessor

- All the known faults of the microprocessor

- The information for reproducing the test cases to verify computational correctness and to have reproduced the known faults

The last item in the list above is necessary when the user needs to reproduce a test case or a fault when testing a system that uses a microprocessor. If any of the above information types are absent for a microprocessor, that microprocessor should be considered as not meeting the level A, B, and C criticality levels.

5.3 MANUFACTURER TRUSTWORTHINESS.

Trustworthiness of a manufacturer is important for predicting the availability of the products in the future and is important when the life cycle of a microprocessor is considered. However, it does not directly affect the safety of a flight or directly cause any failure associated with criticality levels A, B, C, and D (e.g., catastrophic, hazardous/severe-major, major, or minor). Hence, trustworthiness of a manufacturer is not important when considering the criticality levels.

5.4 MICROPROCESSOR TRUSTWORTHINESS.

Trustworthiness criteria of a microprocessor, as discussed in section 4.4 of this report, can be classified in two categories: criteria that do and do not limit the use of the microprocessor in critical applications.

The following trustworthiness criteria of a microprocessor are important for either predicting its lifetime or for improving its confidence level because it has been used in a similar area. Similar to the trustworthiness criteria of a manufacturer, they do not directly affect the safety of flight or cause a failure related with criticality Levels A, B, C, and D. The trustworthiness criteria that do not affect criticality are given below. However, they can significantly affect life cycle costs.

- How long the microprocessor has been in the market?
- What is the expected lifetime of the microprocessor?
- Is the microprocessor used in a similar area?

- Does the manufacturer provide support?
- What point will the support be withdrawn or incur higher cost?
- Is there a warranty and what limits exist on this warranty?

The following trustworthiness criteria of a microprocessor are important when considering criticality levels.

- What is the temperature range in which the microprocessor can operate?

- Is there prompt and automated notification support of updated design/process/die/errata information?

- Is nonmaskable interrupt(s) supported?

- Does the microprocessor have components (e.g., cache, branch predictor) that would cause the execution time to be unpredictable?

- Is there internal address, data, tag, register parity, or ECC protection?

- Is internal or external memory controller/bridge controller supported (requires companion chips)?

- Is there an external interfacing (busses) support for parity protection on address and data?

5.5 PREDICTABLE EXECUTION.

Predictable execution criteria of a microprocessor are concerned with timing anomalies or the difficult to predict upper timing bounds caused by advanced features like the cache, pipeline, branch prediction, and similar functions. A timing anomaly or unforeseen long execution time of a program may result in missed deadline and this in turn may affect the correct execution of the system. High criticality levels require that anomalous behavior do not occur and that the execution time of each program remains within the foreseen limits.

A microprocessor must support features (see section 4.5 of this report) for configuring cache memory such that it does not affect predictable execution. If a microprocessor does not support such configuration options, the WCET time of tasks running on the microprocessor may not be predicted correctly. This may affect the ability to schedule the task set, causing system anomaly. If highly critical tasks cannot be scheduled because of this anomalous system behavior; catastrophic, hazardous/severe-major, or major failures may occur. Hence, if the execution environment of a microprocessor cannot be configured to make the WCET of the running tasks predictable, the microprocessor should be considered as not meeting the Level A, B, and C criticality levels.

For high criticality levels, it is necessary that the unpredictability due to branch prediction and out-of-order execution is eliminated (see section 4.5 of this report). A microprocessor should be able either to turn the prediction off or it should use a branch target buffer for which models are known for bounding the misprediction performance. Similarly, the microprocessor should support configuration options for eliminating out-of-order execution of instruction. If the unpredictability due to branch prediction or out-of-order execution cannot be eliminated for a microprocessor, that microprocessor should be considered as not meeting the Level A, B, and C criticality levels.

5.6 FAULT TOLERANCE SUPPORT.

Fault tolerance support is very crucial for the correct execution of a microprocessor under all circumstances. A microprocessor that does not support fault tolerance for possible error conditions will not be suitable in high-confidence systems. Unrecoverable errors (a bit change in the cache memory that is not detected or recovered) may have dramatic effects such as causing a catastrophic, a hazardous/severe-major, or a major failure. The logic parts of a COTS microprocessor that either deliver the documented performance or are responsible for assuring that the documented performance is delivered should be tolerant to every known fault (including use of approved workarounds).

Section 4.6 of this report discussed a method for assessing the fault tolerance support by considering two case studies. It should be used for ensuring that logic parts of a microprocessor are tolerant to every fault that may occur during flight. If any logic part of a microprocessor that is crucial for its normal execution is not fault tolerant, then that microprocessor should be considered as not meeting the Level A, B, or C criticality levels.

5.7 RELIABILITY OF THE DESIGN.

Reliability of the design is concerned with the effect that the known faults will have on the correct execution of the microprocessor. Availability of the known faults is very crucial as mentioned in subsection5.2 of this report. A fault in a microprocessor design or implementation might affect its correct execution such that it may result in a catastrophic, a hazardous/severe-major, or a major failure. Some design faults may have workarounds and might not affect the correct execution of a microprocessor if a particular configuration is provided. Some other faults may not have workarounds, and they may consistently affect the correct execution of a microprocessor. It is essential that none of the design faults cause any failure associated with high criticality levels (Level A, B, or C). The technique of section 4.7 of this report should be used for ensuring that none of the reported faults will lead to a catastrophic, a hazardous/severe-major, or a major failure. Otherwise, a microprocessor that has the fault should be considered as not meeting the Level A, B, or C criticality levels.

5.8 AVAILABILITY AND SUITABILITY OF TOOLS.

Availability of various tools specified in section 4.8 of this report is important for the development done on the microprocessor. Availability of debuggers will facilitate software development for the microprocessor, whereas the availability of performance analysis tools will ease making timing analysis. However, the evaluation criteria relating to the availability of tools do not directly affect the safety of a flight or cause a failure related with criticality Levels A, B, C, or D. Hence, the availability and suitability of tools are not critical when considering the criticality levels of the target application. However, they can significantly affect development feasibility and life cycle costs.

6. MICROPROCESSOR OBSOLESCENCE.

6.1 INTRODUCTION.

Electronic component obsolescence is a major issue in the avionics industry. However, it encompasses all variety of electronic circuits. This section focuses on microprocessor obsolescence, presents a survey of available approaches, and discusses their advantages and drawbacks.

Microprocessors form the heart and brain of most avionics systems. A typical line replaceable unit (LRU) may contain up to a dozen microprocessors. The typical life of an avionics system is expected to be at least 15-30 years or more. Table 5 shows a few well-known weapon systems and their expected lifetime 5.

Table 5. Life Cycle of a Few Weapon Systems

Weapon System	Expected Lifetime (Years)
F-14	41+
UH-1	49+
F-15	51+
SSN668	56+
AIM-9	72+
KC-135	86+
B-52	94+

However, the microprocessor design, manufacturing, and packaging technology is advancing at a fast and sustained pace, and the trend is likely to continue in near future. For example, the average dynamic random access memory (DRAM) memory cell pitch is predicted to continue decreasing linearly for another 15 years [8]. A similar trend is also suggested for minimum gate length of complementary metal-oxide semiconductor (CMOS) transistors. These two trends are shown in figure 1.

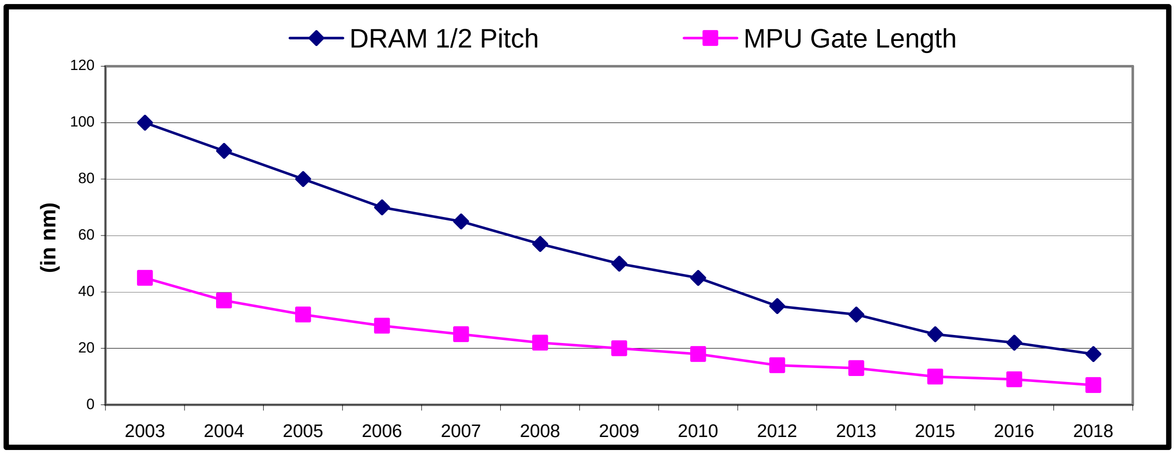

Figure 1. Trend for Minimum Gate Length of CMOS Transistors

Also, transistor density in regular array structure, such as standard random access memory and in general purpose logic chip, is predicted to grow at an exponential rate for the same time period. This trend is shown in figure 2.

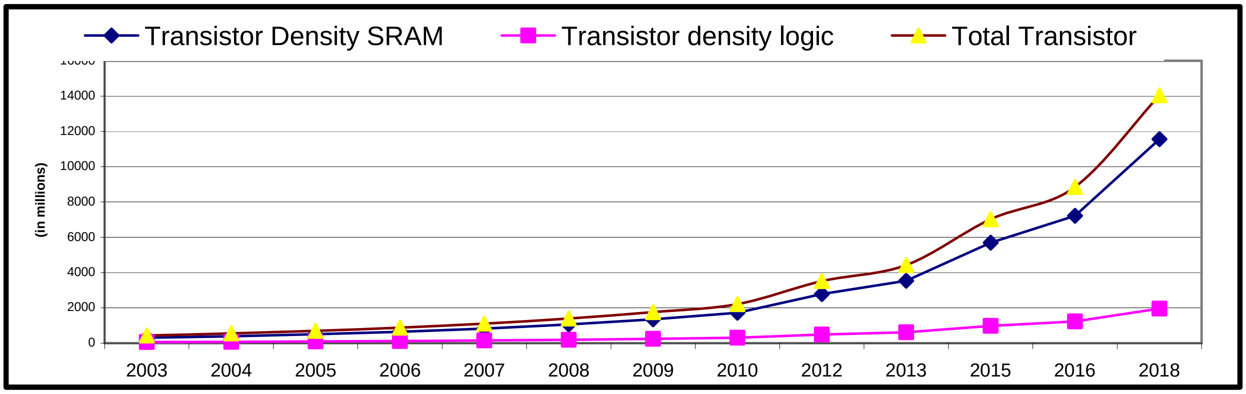

Figure 2. Trend for Transistor Densities on a Chip

The major microprocessor manufacturers frequently incorporate these technological advances to come out with a faster, more efficient, and more capable microprocessor to stay ahead of the competition and to meet the market demand. This results in a fairly short shelf life for the microprocessors that typically range from 3 to 6 years and sometimes even less. Tables 6 through 11 summarize the data for a series of microprocessors manufactured by Intel Corporation. The main reason behind this disparity is that the overall avionics system design, implementation, and certification move much slower than commercial microprocessor design.

Table 6. Lifetime of Pentium Microprocessors

Processor	Speed (MHz)	Start Date	Discontinue Date	Life (Years)
Pentium	100	March 07,1994	November 24, 2004	10.57
Pentium	133	June 01,1995	November 24, 2004	9.35
Pentium	166	January 4,1996	November 24, 2004	8.77
Pentium M	1300	March 12, 2003	April 16, 2004	1.08
Pentium M	1400	March 12, 2003	August 9, 2004	1.39
Pentium M	1500/1600	March 12, 2003	February 18, 2005	1.91
Pentium M	1700	June 02, 2003	February 18, 2005	1.69
Pentium M Low Voltage	1100	March 12, 2003	April 16, 2004	1.08
Pentium M Low Voltage	1200	June 02, 2003	August 9, 2004	1.17
Pentium M Low Voltage	1300	April 07, 2004	February 18, 2005	0.85
Pentium M Ultra Low Voltage	900	March 12, 2003	August 9, 2004	1.39
Pentium Mobile		January 12, 1998	August 17, 1999	1.58
Pentium with MMX	200	January 08, 1997	November 24, 2004	7.77
Pentium with MMX	233	June 02, 1997	November 24, 2004	7.38

Table 7. Lifetime of P4 Microprocessors

Processor	Speed (GHz)	Start Date	Discontinue Date	Life (Years)
P4 Extreme Edition with HT	3.2	November 03, 2003	November 19, 2004	1.03
P4 with HT and 533FSB	2.8	August 26, 2002	April 22, 2005	2.62
P4 with HT and 800FSB	2.4/2.6	May 21, 2003	November 19, 2004	1.47
P4 with HT and 800FSB	3.4	February 02, 2004	March 18, 2005	1.11
P4 with HT and 800FSB	2.8	May 21, 2003	March 18, 2005	1.80
P4 with HT and 800FSB	3.0	April 24, 2003	March 18, 2005	1.87
P4 with HT and 800FSB	3.2	June 23, 2003	March 18, 2005	1.71

Table 8. Lifetime of P3 Microprocessors

Processor	Speed (MHz)	Start Date	Discontinue Date	Life (Years)
PII	450	August 24, 1998	May 5, 2000	1.67
PII	266	May 07, 1997	January 12, 2005	7.58
PII	333	January 26,1998	January 12, 2005	6.87
PII OD for Pro upgrade		August 18, 1997	May 28, 1999	1.75
PII Xeon	400	June 29, 1998	November 19, 1999	1.37
PII Xeon	450	October 06, 1998	February 18, 2000	1.35
PIII	500	February 26, 1999	November 17, 2000	1.70
PIII	533	October 25, 1999	November 17, 2000	1.05
PIII	550	May 17, 1999	November 17, 2000	1.48
PIII	600	August 02, 199	November 17, 2000	1.27
PIII (.18 um)	600E/600EB/650	October 25, 1999	December 15, 2000	1.12
PIII (0.18 um)	500E/533EB/550E	October 25, 1999	November 10, 2000	1.03
PIII (Model 7)	450	February 26, 1999	May 5, 2000	1.18

Table 9. Lifetime of Xeon Microprocessors

Processor	Speed (MHz)	Start Date	Discontinue Date	Life (Years)
PIII Xeon	600/667	October 25, 1999	January 19, 2001	1.22
PIII Xeon	550	March 17, 1999	April 13, 2001	2.04
PIII Xeon	733	October 25, 1999	April 13, 2001	1.45
PIII Xeon	800	January 12, 2000	April 13, 2001	1.24
PIII Xeon	700	May 22, 2000	July 11, 2003	3.09
PIII Xeon	900	March 21, 2001	July 11, 2003	2.27
Xeon	1700/1500	May 21, 2001	August 16, 2002	1.22
Xeon	2000	September 25, 2001	August 16, 2002	0.88
Xeon	2000(533FSB)	November 18, 2002	November 12, 2004	1.96
Xeon (0.13um)	2200(512)	February 25, 2002	March 12, 2004	2.02
Itanium	733/800	May 01, 2001	April 10, 2003	1.92

Table 10. Lifetime of Celeron Microprocessors

Processor	Speed (MHz)	Start Date	Discontinue Date	Life (Years)
Celeron	333	August 24, 1998	November 19, 1999	1.22
Celeron	366	January 04, 1999	January 21, 2000	1.03
Celeron	400	January 04, 1999	May 5, 2000	1.32
Celeron	433	March 22, 1999	July 7, 2000	1.27
Celeron	533	January 04, 2000	November 10, 2000	0.84
Celeron	466	April 26, 1999	October 13, 2000	1.44
Celeron	500	August 02, 1999	December 15, 2000	1.35
Celeron	533	January 04, 2000	December 15, 2000	0.93
Celeron	566	March 29, 2000	April 13, 2001	1.02
Celeron	600	March 29, 2000	June 8, 2001	1.18
Celeron	633/667/700	June 26, 2000	October 12, 2001	1.28
Celeron	2200	November 20, 2002	June 11, 2004	1.54
Celeron	2300	March 31, 2003	June 11, 2004	1.18
Celeron	2000	September 18, 2002	November 11, 2004	2.12

Table 11. Lifetime of Celeron Mobile Microprocessors

Processor	Speed (MHz)	Start Date	Discontinue Date	Life (Years)
Celeron M Ultra Low Voltage	800	January 14, 2003	September 10, 2004	1.63
Celeron M Ultra Low Voltage	900	April 07, 2004	February 18, 2005	0.85
Celeron M	1400	April 07, 2004	February 18, 2005	0.85
Mobile Celeron	266	January 25, 1999	December 17, 1999	0.88
Mobile Celeron	366	May 17, 1999	May 5, 2000	0.95
Mobile Celeron	400	June 14, 1999	July 7, 2000	1.05
Mobile Celeron	433/466	September 15, 1999	July 7, 2000	0.80
Mobile Celeron	400	June 14, 1999	January 19, 2001	1.58
Mobile Celeron	450/500	February 14, 2000	January 19, 2001	0.92
Mobile Celeron	550	April 24, 2000	May 11, 2001	1.03
Mobile Celeron	600/650	June 19, 2000	October 12, 2001	1.30
Mobile Celeron	1800	September 16, 2002	April 16, 2004	1.56
Mobile Celeron	1260	April 16, 2003	April 23, 2004	1.01
Mobile Celeron	1330	June 24, 2002	April 23, 2004	1.81
Mobile Celeron	2000	January 14, 2003	May 21, 2004	1.33
Mobile Celeron Low Voltage	266	January 25, 1999	December 17, 1999	0.88
Mobile Celeron Low Voltage	866	January 14, 2003	April 23, 2004	1.26

The disparity has several major implications on the system design-incorporating microprocessor. The system has to be designed to treat microprocessor obsolescence as an integral activity to design, logistics, maintenance, and evolutionary management activities. The following sections discuss the major approaches towards handling microprocessor obsolescence.

6.2 SYSTEM REDESIGN.

An avionics system incorporating an obsolete microprocessor can be redesigned to incorporate a newer version of the microprocessor. This may cause a ripple of change in the microprocessor board, board support package, Real-Time Operating System, and application software, depending on the extent to which the newer processor has been changed. This can be a very expensive and time-consuming activity due to the effort involved in redesigning and re-certifying the system. Since system redesign may require significant amounts of time, microprocessor obsolescence needs to be predicted in advance to keep production schedule/support impact to a minimum. However, this process can be somewhat mitigated by performing a bridge-buy, which involves purchasing enough microprocessors to support production/repair activity during system redesign efforts. This presents an extra burden to invest in decision support systems incorporating microprocessor obsolescence prediction, system redesign effort, and bridge-buy estimation.

Further, frequent upgrades of microprocessors may cause frequent redesigns of the system. System redesign is a costly activity that has to be managed at multiple levels and evaluated against the cost and complexity of other alternative solutions.

6.3 LIFETIME BUY.

Lifetime buy is a reactive response strategy to deal with microprocessor obsolescence. It involves purchasing a lifetime buy of microprocessors in needed quantities at its phasing-out stage. This appears to be a simple solution to deal with microprocessor obsolescence but has several challenging issues. The most important issue is tracking the microprocessor for obsolescence. Typically, the manufacturer announces its intention to discontinue a product line for a particular version of microprocessor and the associated timeframe. However, sometimes the manufacturer will not give ample notice before discontinuing a product, which leaves very little time for planning and management. There are several agencies (table 12) that offer advanced notification services for products nearing obsolescence.

The second issue is predicting the needed quantities of microprocessors to keep the production/support unaffected for the entire life of the product. This requires a significant investment in data mining tools and decision support systems to predict the inventory. It may not be a permanent solution if the inventory needs are underestimated, which can easily arise if a product or system remains in production/use beyond its estimated life time.

In the case of overestimation there may be significant cost implications in inventory/logistics overhead. For example, in order to handle this situation, national clearinghouses, the defense supply center, etc., invest millions of dollars in buying near-obsolete components that sometimes will never be used. This forces the designers to resort to other solutions.

Table 12. Obsolescence Notification Services

Obsolescence Notification Service	Company Name and Address
ProductWatch	Farnell, Leeds, UK
PrecienceAlert	Precience, Silver Springs, MD, USA
AVCOM	Manufacturing Technology Inc., Fort Walton Beach, FL, USA
PushMail	Government-Industry Data Exchange Program (GIDEP), P.O. Box 8000, Crona, CA, USA
LifeCycle	Synergy Microsystems Inc, San Diego, CA, USA
i2 TACTRAC	i2 Technologies Inc, Dallas, TX, USA
AVLalert, CDS, PCNalerts	PCNAlert, Pasadena, CA, USA

There are also issues with the manufacturer not letting the customer order more than what is available in their current inventory. Manufacturers might sometimes delegate (under some licensing agreement) the manufacturing of certain low-volume microprocessors to small fabrication plants known as secondary suppliers or after-market suppliers. The following is a list of the safety concerns about microprocessors procured from the secondary suppliers. This is also applicable for modified microprocessors that have higher environmental margins in terms of temperature, radiation, and shock rating.

- The integrated circuits' defects are highly dependent on process maturity. Some of the COTS product confidence that comes from statistical advantage of large production volume may not be applicable for low-volume secondary suppliers. Hence, the product may require more rigorous testing than usual for the same product from a large-volume supplier. In brief, due to absence of high-volume production, the fabrication process can not be tuned and optimized for a reliable yield. For further treatment on statistical process control and correlation to COTS component reliability, refer to reference 9.

- The secondary supplier may not use a similar quality standard for fabrication and assembly as the original supplier. Also, the secondary supplier may not have access to the same test equipment as the original supplier. A primary manufacturer, however, can share some of its testing methodology with the secondary supplier to guarantee a level of confidence for its customers.

- The adequacy of the test vectors used by the secondary supplier is also a concern. The problem is somewhat mitigated if the secondary supplier has access to the test vectors and programs used by the original supplier.

- When the secondary supplier switches to new fabrication equipment, how are the processes characterized and what is the yield compared to the original supplier?

- Continuing evolution of interfacing equipment, software, and operations when the components are no longer supported by the primary manufacturers can make continuing use of obsolete components both expensive and hazardous.

6.4 MICROPROCESSOR EMULATION.

An obsolete microprocessor may be replaced with a new microprocessor with a software layer written to emulate the older microprocessor. This ensures binary compatibility and thus the legacy software can continue to run without modification. The microprocessor emulation can be broken down into instruction set emulation, processor-specific functionality emulation, and peripheral emulation. Instruction set emulator is an interpreter for the machine code of microprocessors being emulated. The processor-specific functionality aspect involves the emulation of complex microprocessor functions such as MMU, encryption, decryption, and sound and graphics processing. The peripheral emulations concern the emulation of memory, memory mapped devices, and interrupt handling.

The instruction set emulation is implemented in the form of a loop, which consists of fetching, decoding, and executing the emulated instructions followed by synchronization. A fetch routine uses a software-emulated program counter register to access the next instruction to be executed. The decoding routine extracts the instruction opcode and specified operands based on reduced instruction set computer or complex instruction set computer instruction decoding as required for the machine being emulated. The execution routine maintains a jump table for each type of instruction that consists of code for handling that instruction. The synchronization routine waits for a specified number of cycles to emulate the timing observed in the original processor or to synchronize the operation with external devices to guarantee an error-free operation. A typical emulation loop consists of the following key steps to perform execution of a single instruction of the original processor:

1. Fetching the instruction
2. Incrementing the program counter
3. Extraction of the instruction opcode
4. Fetching the operands
5. Performing a jump to instruction handler
6. Synchronization

The loop may also contain routines to emulate other devices between steps 5 and 6. Since this loop is implemented in software, each of these steps requires a few host processor cycles to execute. The differences in byte ordering between the substitute and original processor can further degrade the performance. Thus, instruction set emulation has a large processing overhead. One study indicated that an emulation layer needs about 20 clock cycles of the new processor in order to simulate a single instruction of legacy processor [8].

Due to this performance critical nature, the emulator is generally written in a low-level language such as C, mixture of C, and assembly or pure assembly, depending on the performance trade-offs. Several other optimizations such as prefetching, branch prediction, and dynamic binary translation can be employed to improve the performance further. The newer microprocessor can be chosen to keep the dissimilarity to the older processor to a minimum. This allows several of the older functionalities such as registers, flags, and specific processing units to be directly mapped to the hardware of the newer processor.

The instruction set emulation must also conform to the timing observed in the original processor to ensure error-free operation. This indicates the need for synchronization within the emulator between various devices being emulated as well as externally to faithfully reconstruct the original processor behavior. To maintain the external timing, a dummy wait cycle is added at the end of the emulation loop if the loop finishes quickly. The number of wait cycles to be inserted is calculated by using experimental methods and tools. The internal synchronization may be achieved by mixing instruction set emulation and device emulations at appropriate frequencies as determined from the system specification of the original processor and experimental methods.

Interrupts and exceptions must also be handled in a timely manner to ensure error-free emulation of the original processor. Sometimes to ensure correct interrupt timing, the instruction loop should contain a routine to enable periodic monitoring or dedicated monitoring, which would otherwise be handled only upon completion of the current instruction fetch-decode-fetch-execute loop.

Memory emulation is a very important aspect of microprocessor emulation and contributes a great deal to the overall performance of the microprocessor. As mentioned above, dissimilar byte ordering (little endian vs. big endian) can place a huge performance overhead on the emulation software due to translation from native to host format and back. Memory emulation has to deal with different kinds of memories present in the system such as random access memory, read-only memory (ROM), and I/O-mapped memories. The accessed memory must be checked to see if the processor is attempting to write in a ROM, which must be adequately denied. Sometimes, software programs are written to include antipiracy measures like writing to ROM to thwart any reverse-engineering efforts. Thus, the correct emulation of each memory type is very essential for the overall emulation. Memory emulation must also check for aligned memory accesses as is required by many central processing units. Memory emulation might also involve emulation of memory subsystems such as MMUs, DMA, bank-switching hardware, virtual-to-physical address translation unit, and caches. All of these subsidiary emulations place some performance overhead on the emulation software.

As emphasized earlier, the emulator is mostly written in assembly language. The software engineering tools for automated testing and validation are usually supported for a higher-level language. This makes testing the emulator difficult. Conformance tests for emulators are generally derived from the manufacturer's specification documents. However, microprocessors may contain undocumented features not officially disclosed. It is not unusual to write programs using the microprocessor's undocumented features. Thus, the microprocessor has to be emulated along with the undocumented features. The problem lies in discovering these undocumented features in an exhaustive manner to derive a complete set of emulation requirements and conformance tests. The testing may also become difficult if a threaded or parallel emulator is implemented. In general, a large number of microprocessor test programs are written and executed on the original processor and an execution trace along with precise timing details using a logic analyzer (or similar testing device) is collected and compared with a similar execution trace obtained for the emulation layer on the target microprocessor. The generation of relevant test programs that provide satisfactory fault coverage requires further investigation.

As noted above, obsolescence mitigation using processor emulation has a large performance overhead and may not be suitable in every case. Further, given the rate of obsolescence, a new microprocessor emulation layer may need to be written frequently, which must be properly certified for airworthiness, leading to a higher amortized cost. Software-based emulation will still have to deal with circuit card assembly incompatibility issues. Another hazard of using a newer replacement microprocessor is the board peripherals that are not required in the LRU being upgraded. Therefore, the unwanted peripherals have to be disabled, put into a safe mode, or monitored if they cannot be disabled in the emulation layer. Finally, there may be legal issues in emulating a copyrighted system, which must be resolved in an appropriate legal framework.

A closer look at the mechanism of implementing emulation software for a microprocessor brings to attention the following safety issues:

- The wait routine needs to accurately measure the number of clock cycles it has to wait before executing the next instruction. An inaccurate measure may lead to external timing problems and may affect the software executed by the emulation layer.

- The performance enhancement strategies such as emulated branch prediction, dynamic binary translation with translation caches, and multithreaded implementation may complicate the calculation of wait cycle on the target processor.

- The unpredictable components (such as cache and pipelining coupled with branch-prediction) in the original processor may inhibit accurate estimation of instruction timing required for accurate calculation of a wait cycle.

- Faithfulness of correlation of the emulated layer with the original processor in terms of test cases and their fault coverage is a safety issue.

- It is highly unlikely that an emulated version of the microprocessor will exactly follow the original microprocessor in terms of timing profile. Discovering a suitable timing granularity in which the performance, functionality, and timing behavior can be shown to be equivalent may be difficult.

6.5 SOFT-PROCESSOR CORE.

The soft-processor core-based solution handles architectural and logic obsolescence. This solution employs reconfigurable logic platforms such as Field Programmable Gate Array (FPGA) to implement a soft-processor core. The soft-processor core separates the computational logic of the microprocessor from hardware implementation. Thus, hardware may continue to evolve independently as long as the interface to the soft-processor core remains intact. Thus, the soft-processor core can be ported to a newer version of the FPGA platform easily or with relatively little effort. This has the least impact on the application software and provides a smooth upgrade profile. Sometimes, a change in the FPGA hardware attributes (such as pin-out, packaging, and environmental and power characteristics) may require board-level redesign [8].

The existing avionics systems containing an obsolete microprocessor may be replaced by its soft-processor version implemented in an FPGA. For example, a Raytheon Missile Bourne Computer Microprocessor has been reverse engineered and redesigned into a soft-processor core hosted in a Xilinx Virtex XCV300-5PQ2401 FPGA using 36K gates. Sometimes an obsolete processor can be accommodated in FPGA along with all its adjoining circuitry, thereby minimizing the number of components and exposed contacts. This increases the overall reliability of the re-engineered system [10]. Please note that this is equivalent to a hardware-based emulation of the obsolete microprocessor. Hence, all the safety issues applicable to software emulation are also applicable in this context.

The reverse engineering of an obsolete microprocessor can be a difficult task due to the complexity of the design and required skill set. The reverse engineering involves accurate cloning of the functionality offered by microprocessors from the information provided by the manufacturer. The manufacturer's data sheet, the schematic diagram, and environmental ratings often omit details, contain errors, and/or unreported bugs and therefore needs to be properly qualified before using them as a reference for reverse engineering. Further, microprocessor design verification involves complex and challenging tasks and requires adequate expertise and investment in sophisticated tools to automate the verification task. Several automated approaches exist which simplify the task of reverse engineering the obsolete microprocessor. These involve an automated test program (ATP) generation to reveal as much of the diverse timing profile as possible [11], and test hardware to automate collection and comparison of the timing profile corresponding to the different ATPs [9].

The soft-processor cores can be classified into generic, original equipment manufacturer (OEM), and third party based on the cost and customization factors. Generic soft-processor cores are often based on some open architecture and are usually available free or at a nominal cost. OEM soft-processor cores have better performance, reliability, development, testing, and verification support compared to generic soft-processor cores. However, OEM soft-processor cores cost a great deal more. There are also third-party soft-processor core offerings that are either modified versions of generic cores or reversed-engineered versions of some popular microprocessors.

VP Technologies has reversed engineered an i860 from Intel, a SHARC from Analog Devices, a PowerPC from Motorola, and a PACE 1750A microprocessors and has further investigated the soft-processor core implementation of the next-generation Motorola PowerPC [12].

For newer systems, Xilinx offers an 8- and 32-bit soft-processor core named PicoBlaze^{TM} and MicroBlaze^{TM} respectively. PicoBlaze occupies a footprint of 35 configurable logic blocks (CLB), runs at 116 MHz, and offers an 8-bit address/data bus. MicroBlaze offers a 32-bit address/data bus that tops at 150 MHz and requires 225 CLBs. Xilinx also offers LavaCORE, a 32-bit Java virtual machine soft-processor core [13]. IBM also offers PowerPC 405 soft-processor core for the Xilinx platform, which can run at 300 MHz and delivers 420 Dhrystone million instructions per second.

7. FEATURE MODELING FOR WCET PREDICTION.

7.1 OVERVIEW.

This section compiles some existing modeling approaches of advanced microprocessor features for making WCET predictions.

7.2 STATIC CACHE SIMULATION.

Static cache simulation was studied in reference 14. It uses compiler-generated information to statically determine the behavior of a large number of cache references (e.g., cache hit or cache miss) prior to execution time of a program. If an address reference can be determined as a cache miss or cache hit with certainty, frequency counters associated with the region of the memory reference are used to describe the cache behavior during the execution time. If the caching behavior of a memory reference cannot be determined with certainty, the static cache simulator is used for discovering the caching behavior during execution.

The address references are categorized as: always hit, always miss, first miss, and conflicts.

Always hit and always miss instructions always result in a cache hit and miss respectively during program execution. First miss category instructions result in a cache miss on the first reference and cache hit on subsequent instructions. The behavior of conflict category instructions cannot be predicted statically (e.g., they may result in a cache hit or cache miss during program execution). The total number of always hit, always miss, and first miss types of instructions is stored in dedicated frequency counters. For conflict types of instructions, state information is associated with the respective code portions, and this state is updated dynamically during simulation. This state information is used for inferring whether the conflict type instructions result in a cache hit or a cache miss.

Static cache simulation modeling does not consider multitasking; hence, WCET unpredictability, due to context switching, is not considered in a static cache simulation. To apply the static cache simulation modeling for a WCET prediction in a multitasking environment, this modeling should be used together with a cache partitioning technique. In this way, each task will exclusively use its own partition and the context switch effects will be eliminated.

7.3 BRANCH TARGET BUFFER MODELING.

Branch target modeling for statically bounding the timing penalty because of incorrect branch predictions was studied in reference 15. The method proposed in reference 15 has some restrictions:

- The maximum number of iterations should be bounded
- There are no recursive calls

- Only conditional statements, loop statements, and function calls can cause breaks in the instruction flow

The method of modeling the branch target buffer for making WCET predictions consists of four steps:

- The program is divided into basic blocks (blocks that do not contain branching), and information about the program's control flow graph and syntax tree are constructed. An abstract buffer state (ABS), of the abstract branch target buffer state is attached to each node of the control flow graph.

- The control flow graph of the program is analyzed to fill in the ABS attached to each node in the control flow graph. The ABS lists the possible control transfer instructions (conditional or unconditional branch instructions) that are in the branch target buffer prior and after the basic block execution.

- The information stored in the ABS is used to classify the control transfer instructions as taken or not taken. The method detects the branch predictions that are guaranteed to be correct. The rest of the branch predictions may be erroneous.

- The classification from above and the program syntax tree are used for computing the worst-case number of incorrect branch predictions. WCET calculation is done in two steps: (1) The WCET is computed based on the assumption that the branch prediction mechanism is perfect (WCETperfect), and (2) the impact of branch prediction is computed separately and added to WCETperfect.

The modeling given in reference 15 does not consider multitasking; hence, the effects of the context switch on the branch target buffer state is not taken into account. For this reason, when a multitasking environment is considered, this modeling should be used together with a branch target buffer-partitioning scheme. Since the branch target buffer is a cache, cache-partitioning techniques can be used.

7.4 PIPELINE MODELING.

The pipeline modeling that this section considers is studied in reference 16. Similar to the techniques explained in sections 7.2 and 7.3 of this report, the pipeline modeling of reference 16 also uses static program analysis for predicting the WCET of programs. Modeling of the pipeline consists of modeling three types of hazards:

- Structural hazards caused by resource conflicts
- Data hazards caused by data dependencies
- Control hazards caused by branches

Modeling of the hazards enables detecting them and subsequently identifies pipeline stalls. Detection of structural hazards requires that the resource usage of each instruction is known in advance. Data hazards occur only when there is dependency between data registers. The read and write ports of data registers are modeled as resources to detect data hazards. The approach in reference 16 proposes that control hazards also be detected with information about resource usage. Resource in this case is the program counter. However, control hazards can be modeled and detected using a more robust approach as explained in section 7.3 of this report.

The modeling defines two states for the pipeline:

- Concrete pipeline state
- Abstract pipeline state

A concrete pipeline state describes the instructions that occupy the pipeline states, the current and future assignments of resources to these instructions, and the state of some special resources like the prefetch buffer. A concrete pipeline state changes whenever a new instruction enters the pipeline. The updated pipeline state depends on the previous state and the new instruction (e.g., the resource demand sequence of the instruction) and the states of other processor elements like the cache (The technique discussed in section 7.2 of this report can be used for modeling the state of cache memories.) An abstract pipeline state consists of a set of concrete pipeline states.

The control flow graph of a program is used for constructing a system of recursive equations. The variables of this system of equations stand for abstract pipeline states from program points. The solution to this system of equations yields abstract pipeline states for control flow nodes. An abstract pipeline state at a control flow *k* gives all concrete pipeline states that may occur when the control reaches *k*.

Once the abstract pipeline state of each control flow is known, the maximal number of clock cycles needed for an instruction to enter the pipeline is determined by simply computing the number of cycles needed by each preceding instruction to enter the pipeline. The maximal number of clock cycles needed to finish all instructions in the abstract pipeline state is determined by computing the maximum number of cycles needed to flush the pipeline.

7.5 PROFILE-BASED PERFORMANCE ESTIMATION.

Profile-based performance estimation is studied in reference 17. This study uses detailed simulators instead of real machines because in real machines it is hard to fix variables and there is no visibility into how exactly the system works. Profile-based performance estimation runs in two stages. In the first stage a benchmark is run once to collect average statistics like runtime of path traces (defined in appendix A), cache behavior, branch prediction behavior, etc. In the second stage the collected statistics are analyzed for estimating the program runtime. The technique is described in appendix A.

8. MICROPROCESSOR TESTING AND VALIDATION.

8.1 INTRODUCTION.

Microprocessors are complex, very large-scale integration (VLSI) devices containing a large number of transistors on a single chip. The number of transistors on single chip is expected to reach 1 billion by 2007, according to reference 18. The advances in silicon manufacturing are further supported by use of innovative architectures to achieve state-of-the-art performance in microprocessors. Examples of such architectural enhancements include on-chip caches, speculative execution, out-of-order execution, memory partitioning, vector computation support, and a host of other features. These technological trends result in considerable challenges for testing and validation of microprocessors. The microprocessor testing and verification has direct implications on the safety of the system incorporating the microprocessor. Hence, it is imperative to understand the various stages and approaches in microprocessor testing and identify the stages and approaches vulnerable to advances in microprocessor design and process technology. The following sections discuss various test stages in microprocessor testing and various test approaches for these stages. The safety and certification issues associated with various test stages and test methodologies are discussed in section 8.4.

The microprocessor testing is performed at several stages during its entire life cycle. The early tests are performed as a part of validation during the development of the microprocessor. The validation tests exploit full knowledge of the internal structure of the microprocessor and are used to ensure the correctness of the microprocessor design. The tests are applied to a simulated design of the microprocessor in a hardware system description language such as Verilog or VHDL. The simulated design is synthesized using manual or automated translation into gate-level description and again tested for ensuring translation correctness. The gate-level description is then mapped into technology-specific, transistor-level description followed by placement and routing to produce a physical-level design. The physical-level design is then verified for crosstalk noise and delay-induced errors. Any errors uncovered in this stage leads to redesign, which is again followed by translation, placement, and routing steps.

After ensuring a certain level of confidence in the design through several iterations, the design is used to derive mask patterns used in manufacturing. The early chips are generated from these mask patterns and rigorously tested using manufacturing test patterns. The manufacturing tests are derived from validation tests, the gate-level description, and an underlying physical fault model based on physics of failure. The manufacturing test patterns are elaborate and designed to uncover and locate the faults in microprocessors resulting from manufacturing processes and correlate the fault with the design. The manufacturing tests are applied using automated test equipment during the wafer sort test stage. The test equipment applies the test patterns at external inputs of the packaged microprocessor and captures the test response at the external outputs. The test responses are verified against the desired response to ensure correct operation of the chip. The chips passing the manufacturing test stage are then put under stress testing to meet its environmental specification. The faulty chips are investigated for sources of errors, which lead to either design modification or tuning process parameters. The chips are further subjected to environmental and class testing to ensure their proper operation under specified environmental conditions.

The fault-free chips are then shipped to the market to be purchased by the intended users. The user validates the microprocessor according to the performance and environmental requirements of the target applications. The qualified microprocessors are then incorporated into the system supporting the intended applications. The microprocessors are subjected to an internal integrity check each time the system is started or restarted. These integrity checks (also known as start-up test) ensure that the system starts its operation in an error-free state. The microprocessors may be subjected to periodic or continuous testing during its normal operation in order to detect runtime hardware failures. These tests are generally low-latency tests and do not perform elaborate diagnostics since most of the computational power must be dedicated to support system tasks such as operating systems and applications. System tests are used to identify the operation of the system and its applications during compliance and regression testing. This necessitates the need for maintenance testing, which is performed periodically after a specified number of hours of operation. The maintenance tests contain criteria for continued worthiness of the microprocessor and suggest a replacement if needed.

8.2 TEST STRATEGY.

The test strategy adopted at any stage depends on the amount of details available about the processor design, test generation algorithm complexity, available testing time, and economy. The test strategy can be divided into structural and functional categories. These strategies are described in the following sections.

8.2.1 Structural Tests.

The structural test strategies are based on the description of the processor design in terms of interconnections of primitive circuit elements such as latches, gates, or transistors. The test generation algorithm assumes an underlying physical fault model to introduce realistic faults in the circuit design. The algorithm then discovers a sequence of input combination, which causes the faulty behavior to manifest itself at the output of the circuit. Each input combination in the sequence is known as a test vector, and the entire sequence is known as a test pattern.

The automatic test generation (ATPG) algorithms generally have to deal with two kinds of digital circuits—the sequential circuit and combinational circuit. The combinational circuits do not contain any memory elements (flip flops or latches), and the test pattern generation is relatively easier. There are several algorithms to automate the test pattern generation for combinational circuits. It is also worth mentioning that some faults may introduce sequential behavior in a purely combinational circuit.

The test pattern generation for sequential circuits (containing memory elements) is very complicated. The test patterns are normally generated and applied in a known memory state. However, the sequential circuit may not be in a known state at the time of testing, which requires either a state identification, complete reset, or an application of an input pattern (known as homing sequence) to bring the circuit into a known state. Exploring the test pattern and homing sequence for a sequential circuit containing large numbers of states is computationally prohibitive, thus restricting the total fault coverage possibly using derived test patterns.

The microprocessors contain a number of different circuits to support different functionalities. Some of these circuits are very large structures and require a large number of test patterns to achieve acceptable fault coverage. The number of test patterns applied is directly proportional to the testing time, an important consideration in mass manufacturing. Hence, the ATPG algorithms attempt to reduce the number of test patterns required to test a given circuit by some suitable partitioning scheme. The ATPG algorithms incorporate partitioning heuristics to automate the circuit partitioning, which can be further aided using structured design to facilitate testability. However, partitioning requires insertion of test points in the circuit to observe its behavior. This adds silicon overhead to the original design. The manufacturer would like to keep the test circuitry overhead as small as possible so that they could use the saved space for improving the performance of the microprocessor. According to Simulation Package With Integrated Circuit Emphasis (SPICE) [19], a 15% increase in die area will result in a construction of new fabrication facility, which has enormous cost implications. Hence, the manufacturer may resort to other techniques to increase the amount of area dedicated to performance circuitry at the cost of other circuitry not directly contributing to the performance enhancement, such as the test area. Hence, the amount of die area dedicated to test circuitry may be an indirect measure of microprocessor reliability.

The structural tests require direct access to the outputs of the circuit, which is not possible in a fabricated chip containing a microprocessor. Therefore, the microprocessor must contain the circuit structures to facilitate the application of test patterns at the input and measure the circuit behavior at its output. The scan test design can be applied to enhance the testability of the circuit. In scan test design, all the memory elements in the circuit have an additional mode of operation known as a test mode. In the normal mode of operation, the memory elements in the circuit do not functionally interface with any testability circuit structures. In the test mode, the memory elements are connected to form the stages of a shift register. Thus, the test patterns can be shifted deeper inside the circuit from the primary inputs and the circuit responses can be shifted from the original output location to the primary outputs. Thus, scan chain design enhances the controllability and visibility and increases the testability for large circuit structures.

8.2.2 Functional Tests.

The functional test strategies are based on the behavioral specification of the processor hardware in terms of memory elements, interfacing signals, interaction, and instruction set semantics. The functional testing can be divided into two main categories. The functional test methods in the first category take the behavioral description and a system level fault model and generate a set of programs to test the microprocessor functionality. The test programs are generated using a signal flow graph derived from the behavioral description. The methods in the second category do not require a fault model and depend on a checking experiment instead. These methods employ test macros, which verify the functionality by sensitizing the Arithmetic Logic Unit, registers, shifter, index hardware, and a host of other functional units sequentially using a set of test programs. The test programs are composed using a variety of approaches. Some approaches use test programs, which contain instructions for all possible operands. A few approaches use a large number of application programs and verify correctness of execution. Since these approaches are not based on the general model of the processor and do not involve any fault model, it is very difficult to correlate the erroneous behavior to physical faults. The following sections describe different functional test program generation approaches.

8.2.2.1 System Graph-Based Method.

The functional tests derived using this method depend on a system graph constructed from the microprocessor architectural specification at register transfer level. The typical architectural parameter required for system graph generation is instruction set, register set, status flags, and interfacing signals. Each register is represented as a node in the system graph. The system graph also incorporates two special nodes IN and OUT to abstract the interfacing with main memory and I/O and other external devices. An instruction may cause a series of data transfers between a set of registers and the IN and OUT node, which can be inferred from the instruction set semantics. The data transfer between two registers caused by a particular instruction is represented as a labeled edge between the nodes corresponding to those registers in the system graph. Load/Store instructions introduce edges between registers and special nodes, IN/OUT respectively. Thus, a system graph represents a high-level data flow model of the microprocessor. The test program generation algorithm further introduces a system-level fault model. The fault model covers the instructions, fetch/decode unit, data storage unit, and data transfer unit. The fault in the instruction fetch/decode unit is assumed to result in an execution of some other statement instead of the specified instructions. The fault in the data storage unit is assumed to result in data value being stuck at zero or one. The fault in the data transfer unit is assumed to exhibit either stuck at or data line shorting faults. The test generation algorithm takes the system graph and the fault model described above and generates test programs, which attempt to uncover the modeled faults. Further details of this method can be found in reference 20.

8.2.2.2 Random Pattern-Based Method.

The functional test approaches require automated algorithms to generate insightful test programs and provide limited fault coverage of about 70% [21]. Please note that 100% fault coverage may not be possible due to inherent lack of visibility of the circuit behavior at the external outputs. Higher fault coverage can be achieved by employing logic synthesis methodologies, which increases circuit visibility at the cost of performance. Pseudo-random patterns are used for generating test programs that contain random instructions. A pseudo-random number generator can be used to generate the instructions that are then fed to the microprocessor under test. The pseudo-random number generator can either be incorporated in the microprocessor or can be simply implemented in the software.

The hardware-based pseudo-random generator generates the instruction opcode and operands randomly and puts them directly on the data bus of the microprocessor. The microprocessor reads the information from the data bus, decodes the instruction, and executes it. The content of each of the registers is verified after the execution of the current instruction is complete. If the verification fails, it is an indication that there is a fault in the microprocessor. Pseudo-random sequences generate illegal opcodes, and there should be a mechanism for replacing them with legal ones by using the last or one of the last five legal opcodes. A filter controls the randomly generated opcodes before they are put on the data bus. If an illegal opcode is detected, the previous legal one replaces it. This requires that the legal opcodes generated by the pseudo-random process are being stored. Please note that this may not be necessary for software-based approaches, which can just ignore the illegal opcodes. The hardware-based random testing is also suitable for the start-up test described below. The software-based approach generates all the instructions beforehand, and then loads the resulting program in memory, which is then executed by the microprocessor.

The random testing can achieve the same test quality as functional testing in a shorter time. The approximate number of instructions contained in the test program is estimated statistically from the level of confidence required that the microprocessor is fault-free. However, the approximation requires the fault distribution data and other statistical measures for estimating the program length. For a mathematical treatment on the appropriate length of the test programs, please refer to reference 21.

8.3 TEST STAGES.

As discussed earlier, the microprocessor testing is performed at several levels during the development phase and at several stages during the entire life of the product. This section discusses in detail the various tests carried out during the entire life of the microprocessor.

8.3.1 Manufacturing Tests.

The manufacturing tests attempt to identify the defective chips early for quality control and cost minimization. Test throughput and coverage is an important consideration at this stage to achieve large-scale production. The test patterns applied during the manufacturing tests are a combination of structural and functional tests derived from a complete knowledge of the processor design. The tests are generated using ATPG and functional test generation algorithms. The tests are elaborate enough to enable diagnostic evaluation to correlate the faulty behavior to a defect in the design or the manufacturing process. The microprocessors incorporate scan-based test structures to facilitate the delivery of the test patterns and the measurement of response. Automatic test equipment is used to deliver the test patterns and verify the response. The scan registers are tested first by selectively enabling them while in the test mode with the combinational part of the circuit operating normally and then shifting a known test pattern through the scan chain. The output is measured at the external pins and compared against expected results to ensure the correct operation of the scan chain operation. The rest of the circuitry is then put into test mode and manufacturing test patterns are applied and measured response is verified against an expected response. The test of the microprocessor production patterns are applied at different packaging levels, starting from wafer level to the diced and packaged level. The manufacturing tests are also applied at the assembly level when the microprocessor has been incorporated on a printed circuit board.

A drawback of the scan circuitry is that internal output nodes respond continuously to each test pattern applied. This results in a continuous toggling of the output node voltage level and is a source of excessive heat generation. The cumulative effect of the excessive heat generation across all output nodes may be sufficient to cause metal migration or melting of the test structure if the tests are carried out at very high clock frequencies.

The manufacturing tests can be divided into the three main stages (wafer sort, environmental, and class tests), as described in the following sections.

8.3.1.1 Wafer Sort Tests.

The wafer sort test is used to identify and separate the defective die at the wafer stage to reduce packaging costs. The wafer sort test is normally performed before depositing the second metal layer. This ensures that enough electrical contacts are available for thoroughly examining the logic devices. The wafer sort test may also employ some additional tests such as differential Hall effect and secondary ion mass spectrometry. The manufacturing test patterns are delivered through the flying probes at normal room temperature. However, it may be economical to perform a limited environmental test to identify more defective chips early in the production. One such trend is to sort the wafers at a lower temperature. This is also known as cold socket examination and requires the microprocessors' wafers to be kept below 35°C during the wafer sort test. The typical duration of these tests range from hundreds to thousands of milliseconds and depends on the capability of the test equipment and the complexity of the device being tested (i.e., microprocessors).

8.3.1.2 Environmental Tests.

The environmental test ensures the correctness of devices when subjected to extreme temperatures in the specified temperature range and other environmental stresses. The environmental tests consist of burn-in tests, thermal stress, vibration stress, pressure tests, and a host of other tests. The burn-in test involves putting several packaged devices (several thousand, depending on capability of test equipment) at elevated temperatures and voltages to speed-up the reliability-oriented defects arising out of process imperfections. The amount of time the device is subjected to the burn-in test depends on various factors such as yield, failure rate, voltage, and junction temperature. The elevated voltages tend to accelerate the burn-in faster than the elevated temperatures; hence, they are kept as high as possible. Typically, the elevated voltages go up to 1.4 times the rated voltage during the burn-in test without damaging the device. The typical amount of time the devices are subjected to the burn-in testing range from 10-168 hours for microprocessors, with the upper limit for microprocessors conforming to military specifications. The thermal shock test involves subjecting the device to high temperatures and then quickly moving it to low temperatures. The humidity-based tests can be used to assess the impact of corrosion. The test involves placing the microprocessor under extreme humidity and temperature conditions. The stress tests are specified by a pattern that is the value of the attribute (temperature, voltage, pressure, or humidity) as a function of time.

8.3.1.3 Class Tests.

The class test forms the last stage in the manufacturing tests. The packaged devices in this stage are again subjected to ATPG-derived test patterns to validate the processor's functionality. In addition to this, an additional set of tests is performed to determine the speed of the processor in the specified temperature range. The processors are sorted and placed into different class speed bins. Hence, the class test is sometimes referred to as speed binning. The microprocessor speed is dependent on the operating temperature. A typical estimate is that a microprocessor speed decreases by 0.15% for each additional degree centigrade in the operating temperature [22]. Therefore, precise control of the operating temperature during the class test is very important. However, due to the very high-frequency operation and the trend towards highly integrated processors, the chips exhibit a nonuniform heating pattern that is increasingly becoming a concern in class testing.

8.3.2 Qualification Tests.

Qualification tests are used to evaluate the suitability of a microprocessor for a specific application. The qualification tests have two main parts: part one deals with validating information provided by manufacturer, and part two deals with assessing performance parameters for the intended applications. The qualification test may include part of manufacturing tests provided by the manufacturer under a nondisclosure agreement. The qualification tests may also include environmental tests to validate the thermal and mechanical specifications of the microprocessor. The microprocessor functionality is validated by applying functional tests derived in the user environment. These functional tests can be generated using the techniques discussed in section 8.2.2. The tests also validate various timing and power specification provided by the manufacturer.

8.3.3 System Validation and Maintenance Tests.

The end system containing the microprocessor is validated using a variety of approaches. These approaches use a large variety of random application programs to achieve good statistical test coverage. Some approaches additionally inject known microprocessor faults during its operation to ensure correctness. Sections 8.3.3.1 and 8.3.3.2 discuss these approaches.

The system validation tests can also be used as maintenance tests. The maintenance tests are periodically performed after a specified number of operational hours. The exact number of hours is determined from the empirical data. The maintenance tests are also needed after a hardware, software, or firmware upgrade.

8.3.3.1 Operational Tests.

The operational tests ensure the functional correctness of the microprocessor by running a large variety of application programs. The applications range from compilers, operating systems, and actual programs intended to be run on the processor in the final system. The operational tests use the microprocessor and its peripherals in conjunction. The other tests mentioned above test the microprocessor in isolation. The operational tests run at the speed of the microprocessor and test the interaction of peripheral components in real time. Since operational tests are performed on the final system, they are also known as system tests. These tests judge the overall correctness of the system built around the microprocessor. However, the operational tests tend to be large and exhaustive and consume lots of time. The time between two critical test conditions is large due to general programs employed for testing. Further, the test failures cannot be correlated with actual microprocessor faults. The operational tests cannot be used for stress testing the system. For example, an ad-hoc, burn-in test approach in the user-environment employs complex programs that are known to be power-hungry. However, such programs may not be able to achieve the thermal stress pattern (demanding fast temperature changes) required to validate the environmental correctness of the microprocessor operation.

8.3.3.2 Fault Injection.

Fault injection (FI)-based validation is done through deliberate insertion of faults during the system operation and validating its response. FI can be done using a software-implemented technique, a hardware-implemented technique, or a combination of the two. Hardware-based techniques are capable of injecting more sophisticated faults in the system. The FI environment specifies all the attributes necessary for injecting faults into the system being verified for correctness. The FI environment is characterized using the FARM classification scheme, which is comprised of the following attributes:

- Faults: The set of faults that are injected into the system.

- Activation Trajectories: The input domain used for functionally testing the system.

- Readouts: The observed behavior of the system.

- Measures: Dependability measures obtained through the FI experiment. They are obtained experimentally from a set of case studies.

The challenge with FI-based verification of a processor is that injecting a fault might affect the execution environment or the runtime of the target system. This phenomenon is known as FI intrusiveness. Intrusiveness is caused by different reasons:

- New instructions or modules can be introduced for supporting FI. Hence, the set of executed instructions and modules are different than when the FI is not used.

- FI may change the electrical and logical setups of the target system affecting its execution speed. This phenomenon is called time intrusiveness.

- Introduction of new code for supporting FI can change the memory image of the target system.

Ideally, when FI is done, intrusiveness should be minimized. Otherwise, it is not possible to extend the results obtained from the FI experiment to the original target system. Different types of faults can be injected in the execution environment of a system. Common faults include transient single-bit flip, bridging faults, multiple-bit flip faults, and stuck-at faults. The study in reference 22 injected illegal, reserved, and undefined instructions for the processor being tested.

The study done in reference 23 used transient single-bit flip type faults. Each fault is characterized by the following information:

- FI Time: Each fault is injected into the assembly level before an instruction is executed.
- Fault Location: A fault is injected either into a memory location or into a register.
- Fault Mask: The bit mask that specifies the bit(s) that will be flipped.

8.3.4 Start-Up Tests.

The start-up tests are very important in the safety-critical systems, and they ensure that the microprocessor begins operation in an error-free state. The system start-up can be differentiated into cold start or warm start, depending on whether it is started from an off state or from an on state. The cold start generally happens when the system is started for the first time or after a period of inactivity, and the microprocessor will require rigorous testing due to the inactivity. The cold start-up tests are therefore allocated more time than warm start-up tests to allow for the elaborate testing. The tests in this stage can be done using a relaxed fault model compared to manufacturing tests. The microprocessor should aid the system diagnostic module to report any faulty behavior. The warm start-up generally happens after a run-time exception to the whole or a part of the operational system.

8.3.5 Monitoring Tests.

The monitoring tests are also important in the safety-critical systems. The monitoring tests ensure that the microprocessor is operating correctly by checking the processor either periodically or concurrently. These two techniques are described in the following sections.

8.3.5.1 System Executive/Safety Executive.

The monitoring tests can be implemented by a separate controller system known as the safety executive. The safety executive is responsible for periodically testing the system's health and the operational correctness. The system executive can be implemented in software and can test the microprocessor during periods of inactivity. Some systems use a watchdog timer to implement the system executive's functionality. Since the computing resources are limited during monitoring tests, most of the techniques use cyclic redundancy check (CRC)-based checking to detect run-time fault. For example, some languages may associate a CRC with each data structure that may be verified by the system executive.

8.3.5.2 Built-In, Self Test.

The BIST mechanism in a processor is responsible for checking the processor functionality concurrently with system operation. Since the run-time performance of the processor is a crucial factor, the BIST mechanism employs error detection strategies that cause minimum intrusiveness in the system operation. Thus, the error detection logic depends on CRC checksum or a more sophisticated coding scheme to uncover faulty behavior. Thus, data on bus, on-chip memories, and register contents might have a CRC checksum that is monitored by a self-test mechanism and flags errors as soon as they occur. Part of the BIST circuitry performs very elaborate tests and is activated only during the system start-up. Thus, the BIST mechanism provides limited functionality for monitoring tests.

8.4 SAFETY AND RELIABILITY ISSUES IN MICROPROCESSOR TESTING AND VALIDATION.

The trend towards high-performance microprocessors and the existing approaches toward testing reveal some interesting safety and reliability issues in safety-critical computing for adoption of these microprocessors. These issues are discussed in the following paragraphs.

The manufacturing tests are applied at the core frequency using automated test equipment as previously described. The core frequency is steadily increasing in the state-of-the-art performance microprocessors that result in requiring faster test equipment. Further, timing variation introduced due to submicron effects such as cross-coupling and inductive noises make timing verification extremely difficult. These issues impose a greater performance requirement on the test equipment that must operate must faster than the device under test. Thus, the test equipment required for the next generation of microprocessors tends to be very costly. This has serious effects on testing microprocessors used in safety-critical system designs. A survey indicated that some companies in the avionics industry depend on the manufacturer-supplied test patterns to verify and validate the microprocessor functionality. However, this approach may become infeasible due to the cost of the automatic test equipment required to deliver the test patterns at the core speed.

Another issue worth mentioning is the environmental testing of the microprocessor. Due to increasing core frequency and trends toward highly integrated microprocessors, the temperature gradient of the microprocessor core may not be uniform. This requires microprocessor manufacturers to resort to very advanced techniques to control the devices' temperature for class testing and for improving the yield of high-speed bins. However, the rated performance cannot be readily verified without the ability to have the same advanced test equipment and techniques as those used by the manufacturer.

One more interesting trend revealed from the survey indicated that most of the avionics companies use application test cases developed as part of the DO-178B to qualify the microprocessor. However, research suggests that these ad-hoc testing approaches may not achieve the desired statistical test characteristics required for safety-critical systems. The literature suggest much more effective approaches are needed to achieve higher confidence levels in the microprocessors using alternate test methods that employ system-level functional fault models.

On-chip caches on microprocessors constitute very high-density structures occupying about 65% to 80% of the die area. Registers and data paths consume most of the remaining die area. These structures require very elaborate fault models and design efforts to enable their testing. Some of the faults may also be very hard to model, leaving a portion of these structures untested, which may have implications on their reliability. To increase the test reliability, several strategies such as design-for-test, BIST, and redundant structures have been adopted. However, the extent to which these techniques are used in practice is decided by the corresponding die area overhead.

For example, a Motorola Cold Fire processor allocates about 5% of total die area for dedicated test circuitry. Also, there is an increasing trend to squeeze this area as much as possible to boost the processor performance due to commercial interests.

In view of these test strategies and trade-offs adopted by industry in microprocessor testing, there is concern about its reliability in safety-critical systems. Caches are, in particular, a cause of concern as it introduces probabilistic behavior in execution time of a process. Normally, this probabilistic behavior can be bound by a combination of partitioning and rigorous WCET modeling techniques. But, all these techniques and models assume the caches to be reliable and undegradable over time. However, as described earlier, the caches are very high-density structures, and some of the cache faults are difficult to model. This leads to an unacceptably high probability of residual errors and/or susceptibility to manufacturing defects. For this reason, most of the processor designs include redundant cache blocks and dynamic self-repair by modifying the selection path logic in combination with some error flags. If the defect is detected at the time of manufacture, the caches can be repaired using an ion beam to use the redundant blocks. These types of faults can be detected during system testing on a target platform to validate the WCET models for various data paths. However, if the defects are detected later during the processor operation, processor self-repair logic selectively disables the faulty cache blocks. For example, the self-repair logic may include a redundancy register, which is loaded during processor start-up self-test. The self-test logic may discover a faulty cache block and disable it by appropriately modifying the redundancy register. This type of error may remain undetectable during the system certification and can pose a serious safety risk during actual system operation. Moreover, the processor may not incorporate a mechanism to query the amount of currently available cache memory to implement the safety mechanism that will signal such operational errors.

9. MICROPROCESSOR INTEGRATION IN SoC.

9.1 INTRODUCTION.

The modern semiconductor fabrication technology is capable of manufacturing devices containing hundreds of millions of gates. This creates the possibility of integrating full digital systems on a single chip for increased performance and low-cost manufacturing. To promote rapid prototyping and integration of digital systems on a single chip, several vendors offer pre-tested and pre-verified common logic components known as Intellectual Property (IP) Cores. IP cores reduce the overall system development by promoting reusability.

This trend is also embracing the microprocessor industry, which is increasingly offering highly integrated microprocessors to a sea of microprocessors on a single chip. In highly integrated microprocessors, the peripheral devices supporting a microprocessor are fabricated on the same die. Example of such peripheral devices include Ethernet controller, fabric interface, memory controller and I/O interface, and a host of other IPs. In addition, the SoC approach allows the customization of microprocessor design parameters such as cache size, floating point units, and pipeline structure to support a specific application for a highly efficient design.

However, the integration of different IP blocks (peripheral devices and other application-specific system functionality) raises signal integrity issues due to the close proximity of the components, increasing metal layers, and high-frequency system operation. Also, the shrinking feature sizes and the low power design make the circuit susceptible to power grid fluctuations and becomes a major issue. The power grids can also be susceptible to electro-migration problems when the power supply noise margin is not properly met. These problems require a careful and detailed analysis of the IPs being integrated and causing possible interference that would affect signal integrity, power grid noise margin, and any other incumbent issues. The analysis is more complicated if the IP blocks are obtained from different vendors that have been analyzed and tested in the absence of any neighboring blocks that might be present in the current SoC design.

9.2 INTELLECTUAL PROPERTY CORES.

IP cores, also known as virtual components, allow implementation of highly efficient systems. The system designer can select the most appropriate IP core that meets a particular performance requirement among several similarly available IP cores. The IP cores are specified and exchanged in industry standard format with varying level of details. The IP cores are generally classified in the following three categories according to varying level of details:

- Soft IP core
- Firm IP core
- Hard IP core

The soft IP cores contain the maximum level of detail and are generally specified in register transfer level description in languages such as Verilog or VHDL. Thus, soft IP cores allow detailed analysis and optimization of the system being integrated. The firm IP cores are next in the decreasing level of detail and specified in technology-independent netlist level format. This allows the IP vendor to hide the critical IP details and yet allow the system integrator to perform some limited amount of analysis and optimization during placement, routing, and technology-dependent mapping of the IP block. The hard IP cores are the lowest in level of detail and specified in technology-dependent physical layout format using industry standard languages such as stream, polygon, or GDSII format. The hard IP cores are like black boxes and cannot be properly analyzed and/or co-optimized. Hard IP cores come with a detailed specification of integration requirement in terms of clock, testing, power consumption, and host of other parameters.

For the purpose of discussion of integration issues, this report will also categorize IP cores as homogenous or heterogeneous IP core, depending on whether or not they come from the same vendor.

9.3 INTELLECTUAL PROPERTY INTEGRATION ISSUES.

In traditional systems, the different logic components are usually implemented in different chip-level packages and connected in a board-level integration. The distances between different components in a board-level integration is large enough to make any mutual interferences affecting the integrity of signals present on the VLSI interconnects inside the chip negligible. However, when these components are integrated on a single die, the distances between the different components are of submicron order, which brings in various VLSI interference issues such as crosstalk noises and power-grid issues.

9.3.1 Crosstalk Noise.

Coupling capacitance between two adjacent nets, which creates a temporary electrical path during signal transition, causes the crosstalk noise. Here, a net is referred to as an electrically equivalent group of wires. The net carrying the signal under transition is called an aggressor net. The net being affected due to signal transition in another net is referred to as a victim net. The amount of injected noise is directly related to signal transition time, also known as the slew rate, and the coupling capacitance, which is dependent on process technology and effective overlapping of wire segments forming the nets.

The crosstalk noise affects the functionality of the circuit in two ways. First, crosstalk noise can cause a small glitch in the victim net that is subsequently propagated to a latch. This causes the state of the circuit to change and affects the overall functionality. This type of noise results in a functional error and is referred to as functional noise. The second type of noise is manifested when two cross-coupled (due to coupling capacitance) nets switch simultaneously. This may either increase or decrease the delay of both the nets. If the victim net experiences an increased delay and is present on the critical maximum path delay, it results in a setup failure where the signal arrives too late to cause the desired switching. If the result of the switching is expected to result in a state change, the circuit will enter into an erroneous state. Similarly, if the victim net experiences a decreased delay and is present on a critical minimum delay path, it will result in a hold failure, again possibly affecting the state of the circuit. This type of noise results in setup/hold failure and is referred to as delay noise. The delay noise can also manifest in another form where several small path delays (too insignificant to cause any errors themselves) have a cascaded effect. The cascaded delay of several nets present on a critical delay path has a cumulated effect on setup/hold failure. This type of noise is hard to analyze compared to the scenarios described above.

As the industry adopts the newer manufacturing processes with smaller feature size and higher frequency of circuit operation, the ratio of coupling capacitance to parasitic capacitance is increasing. The most important scaling factors resulting in an increasing coupling capacitance are as follows:

- Minimum Metal-Metal Separation
- Reduced Metal Pitch (0.36 μ in 0.18 μ process as compared to 0.26 μ in 0.13 μ process)
- High Slew Rate (High frequency of operation)

These factors become dominant with each scaling down of the manufacturing process, making crosstalk noise more dominant. In the latest process technology, it is estimated that the crosstalk noise may affect the delay of a signal by more than 100 percent. It may be noted that the increased number of metal layers will not affect crosstalk noise significantly as the adjacent metal layers act as shields for other higher layers.

The crosstalk noise analysis involves the extraction of coupling capacitance data and parasitic-capacitance data for the entire circuit. The three-dimensional capacitance extraction tool such as space3d can automate the coupling and parasitic-capacitance extraction task from GDSII physical layout of the IP cores. A typical microprocessor core may contain hundreds of thousands of nets, prohibiting circuit simulators such as SPICE [19], which can handle thousands of nets and requires hours of simulation time (a typical microprocessor core usually has 227,000 nets). Hence, a reduced order circuit model is obtained under reasonable assumption. The circuit is further pruned out to eliminate the nets whose glitch noise remains under a particular threshold. The resulting circuit is analyzed for functional and delay failures. Further details of crosstalk noise analysis can be found in reference 24.

The IP cores are usually preverified and pretested and considered to be clean from any signal integrity violations. However, the timing analysis is usually done without any information about the adjacent IP cores that will be present in the final system. The neighboring IP cores in the final system may introduce new signal integrity violations due to coupling capacitances between wires of different cores routed close to each other. The unexpected noise injected into an IP core by its environment may not have been anticipated by the IP core designer, which may affect the signal delay on critical path and lead to noise-induced failure.

Some IP cores permit over-the-block routing where the global IP core interconnects use the higher metal layers present on top of the IP core block. The over-the-block routing may interfere with the top metal layer of the core, although the lower layers are usually shielded. The effective overlap between the global interconnects and the top metal layer of the core may introduce significant noise into the core.

The effect of such interaction can be analyzed if the integrator has sufficient visibility into IP cores in terms of electrical and physical on-chip interconnect properties and logical and timing window data. This may be possible in case of soft or firm IP cores where these data can be extracted in sufficient detail to permit accurate analysis. An alternative to this complete reanalysis of the whole system is for the IP core designer to share data about the critical set of signals and nets. IP core designers can also specify the coupling and shielding constraint on the design. The integrator takes these constraints into account during the physical layout and routing of the cores and interconnects.

9.3.2 Substrate Noise.

Substrate noise is caused by capacitive coupling of wells, which creates a temporary electrical path during signal switching. The created temporary electrical path will contain small amounts of alternating current that may affect any adjacent analog component or will be propagated through on-chip interconnects to other analog components. The substrate noise results in performance degradation or failure of analog components present in the SoC, if they are integrated on the same die. Examples of such systems include radio frequency cores, analog-to-digital data converters, and memory blocks (sense amplifiers, etc.).

9.3.3 Power Grid.

At a submicron level, another issue that affects the integration of IP cores is IR drop (I-current times R-resistance) along a power-grid. Due to lower device voltages and increased power consumption, the supply voltage is continuously increasing to support newer process technologies resulting in a significant IR drop. The IR drop problem is anticipated to be increasingly prevalent in the 130 nanometer and below process technology.

IR drop can affect the performance of the system in several ways. For example, the IR drop can alter the effective voltage, V_{DD}, being supplied to transistor devices. The dynamic variation in V_{DD} can sometimes go out of the specified V_{DD} margin, which may affect the performance characteristics of the transistor, resulting in a functional failure. IR drop can also cause increased gate delay, resulting in timing failures. These failures may cause a state change in the circuit and may result in an unpredictable circuit response leading to an eventual system failure.

The dynamic voltage variation may be caused by submicron inductive effect or simultaneous digital switching activity straining the global power supply. The inductive effects are largely dependent on the transient signal timings such as rise or fall time. Thus, as the system design moves towards the gigahertz scale, the inductive effects become a significant factor of overall IR drop.

The dynamic voltage variation may also cause the current density of some metal wires to be significantly above the rated maximum current density. The higher current density may accelerate the electro-migration process that causes a metal short or metal breaking.

10. SUMMARY.

Both ground and airborne avionics are dependent upon the use of COTS microprocessors. Evolving microprocessor architectures include concepts such as caching, pipelining, branch prediction, and other advanced features that can affect system performance, predictability, and safety. FAA qualification and certification policy and procedures of safety-critical avionics require proof of safety. There is a risk of these regulatory activities becoming exorbitantly expensive and less effective in ensuring safety requirements are met. Microprocessors and other complex hardware (e.g., (SoC)) are driven by market forces that generally do not consider avionics safety requirements and are rushed into production to meet competitive goals. Due to the need to (1) select COTS components that are less expensive, (2) meet evolving system requirements, and (3) select components that are still being produced and maintained by the manufacturers' designers, integrators, and maintainers are forced to use the ever more complex hardware. In most cases, the detailed design and test information held by component manufacturers are not available due to the competitive, proprietary nature of the market. Microprocessor manufacturers are not designing or testing their products to meet safety-critical application requirements because of the cost and potential exposure to litigation.

The following three categories of aerospace systems reflect the growth of complexity related to the timing of the related certification effort.

- Systems that have been certified and are currently being maintained
- Systems that are currently undergoing certification
- Systems that are going to incorporate emerging features that are more complex and less deterministic

As the components are increasingly complex, progressing from past to present to future, each category has an increasingly complex set of issues. All three categories exhibit nondeterministic characteristics. The trend seems to indicate an ever increasing cost of safety evaluations and a divergence between existing regulatory policy and the characteristics of the systems being certified.

The intent of this phase of research was to provide findings about safety issues in using modern microprocessors on aircraft. The research considered the applicability of DO-254 to all three categories of avionics systems listed above, documented potential safety concerns when using modern microprocessors on aircraft, and proposed potential approaches for addressing those safety concerns. In Phase 1, avionics systems developers (BAE Systems, The Boeing Company, and Smith Aerospace) joined with the FAA in sponsoring this research at Texas A&M University to establish the scope of the project and identify research parameters as documented in this report. This project provided the opportunity to integrate industry approaches into using the evolving microprocessor technology in future avionics and safety-critical applications with the FAA's evolving regulatory policy, guidelines, and procedures for safety within the qualification, certification, and acceptance processes.

10.1 FINDINGS.

Phase 1 revealed that the safety analysis of continually evolving modern microprocessors in aeronautical and space application has received little attention from research and analysis efforts. The trend towards nondeterministic complexity of COTS microprocessors and the resultant increase in cost and chance of unassured safety in system certification indicates significant risk to a vital part of the U.S. economy. The research may result in significant savings to both the FAA and certifying applicants due to the identification of accepted methods for safety analysis of the complex hardware in avionics systems and the resultant simplification of the certification processes. There is also a high probability that the risk of unassured safety related to complex microprocessors can be mitigated by the results of this research.

Due to competitive market forces, both (1) microprocessor design and test information held by each microprocessor manufacturer and (2) the certification approaches and methods developed by individual certifying applicants are considered proprietary and thus are partitioned from each other. This situation is further obscured by existing certification policy and procedures by requiring that each certification stand alone. This results in the repetitive design, proof of failure mode information, and the evaluation of the safety evidence approach for each certification of complex nondeterministic hardware (e.g., microprocessors). There is currently no single source of information being maintained for safety analysis methods and techniques related to the certification of these complex systems and their potential failure modes.

This report presents an assessment of existing certification guidelines towards certification of evolving microprocessor architectures and emerging features. It indicates that new certification processes are required in addition to the existing ones. This report also addresses unpredictable issues in computational components of the microprocessors that may lead to safety concerns in avionics applications.

The safety issues are raised when using a microprocessor in avionics products. The issues are primarily due to the lack of trustworthiness of manufactured components, unavailability of sufficient information on microprocessor designs, unpredictability of run-time tasks due to complex features in the microprocessors, and insufficient test and fault-tolerant support within the microprocessor systems. The report provides the suitability, unsuitability, and safety concerns regarding DO-254 for evaluating microprocessors. A list of possible evaluation criteria (such as availability and validity of information, trustworthiness of a microprocessor, predictable execution, fault tolerance support, reliability of the design, availability and suitability of tools, adequate mitigation for microprocessor obsolescence, and performance and functional testing) was identified to provide a basis for the development of standards, guidelines, and processes to be used to augment the guidance of DO-254. A tentative classification of criteria in safety levels A through E based on the level of criticality (from higher to lower) was given. This report identifies that microprocessor obsolescence management may become a significant problem in the future due to rapidly changing designs. Feature modeling is going to be a potential approach to identify risk involved in a feature and then the microprocessor as a whole. The microprocessor testing and evaluation trends are presented and several safety concerns are identified relating to tests, qualifications, and certifications. The report also outlines the risk-related issues that arise in a SoC that integrates microprocessor IPs with peripheral components (i.e., memory controller, bus controller, and interrupt controller) or multiple IPs on the same chip (including multiple processor cores).

10.2 RECOMMENDATIONS.

Based on the findings of Phase 1 and the significance of the certification process, the following are identified as recommended research activities. These activities are intended to further research and to possibly provide support to both regulatory agencies and industry.

- Identify ways to augment existing regulatory policy or establish new policy to provide procedures and methods to permit the safe, economical qualification of microprocessor applications with complex nondeterministic architectures.
- Develop guidance for field service credits that can be achieved for existing and derivative microprocessors.
- Identify microprocessors and/or microprocessor architectures and features for safety-critical aerospace applications that can be proven to be safe.

- Establish acceptability criteria for instruction set architectures that are incrementally evolving or stay constant in different families of microprocessors.

- Balance the FAA regulations and policy with the design and test of microprocessor components and systems that display nondeterministic characteristics.

- Determine how to economically and realistically test modern, complex microprocessors and related avionics systems to meet safety requirements.

- Identify probable faults in a manner that limits test and evaluation efforts and optimizes the likelihood of meeting safety requirements.

- Identify and/or provide tools to support the qualification and certification processes and/or to provide safety evidence.

- Examine methods to evaluate nondeterministic complex microprocessors and components containing microprocessors (e.g., SoC).

- Determine possible ways to maintain and share failure mode information and safety evaluation methods and techniques to facilitate growth and viability of aeronautical and space industries.

These activities should be directed primarily to the second category of systems identified above. As Category 1 systems undergo maintenance, logistics will require the replacement of components with those microprocessors that are within Category 2 applications, and this research and the resultant methods and processes will be applicable to them.

The emerging features inherent in Category 3 systems and their applications should be a subject for research following Phase 2. The size and complexity of the required research is beyond the resources of the AFE number 43 Microprocessor Evaluation Project as it is currently funded and contracted.

11. RELATED DOCUMENTATION.

Birdsong, B. and Walter, K., “Criteria for Selection of COTS Equipment in a Military System,” *Proceedings of the 2002 DMSMS Conference*, 2002.

Alford, L.D., “The Problem With Aviation COTS,” *IEEE Aerospace and Electronic Systems* Magazine, 2001, pp. 33-37.

Hennessy, E., “COTS Supportability Across the Program Lifecycle: A Look at the COTS Evolution in the DOD Market,” Sky Computers Inc., 2000.

Carney, D.J. Morris, E.J., and Place, P.R.H., “Identifying Commercial Off-The-Shelf (COTS) Product Risks: The COTS Usage Risk Evaluation,” Technical Report, CMU/SEI-2003-TR-023, ESC-TR-2003-023, 2003.

Bate, I. Conmy, P. Kelly, T., and McDermid, J., "Use of Modern Processors in Safety-Critical Applications," *The Computer Journal*, Vol. 44, No. 6, 2001, pp. 531-543.

Vera, X., Lisper, B., and Xue, J., "Data Caches in Multitasking Hard Real-Time Systems," *Proceedings of the 24th IEEE International Real-Time Systems Symposium*, 2003, pp. 154-165.

Basumallick, S. and Nilsen, K.D., "Cache Issues in Real-Time Systems," *ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Real-Time Systems*, 1995.

Lundqvist, T. and Stenstrom, P., "Timing Anomalies in Dynamically Scheduled Microprocessors," *Proceedings of the 20th IEEE Real-Time Systems Symposium*, 1999, pp. 12-21.

Campoy, A.M., Ivars, A.P., and Busquets Mataix, J.V., "Using Locking Caches in Preemptive Real-Time Systems," *Proceedings of the 12th IEEE Real Time Congress on Nuclear And Plasma Sciencies*, 2001, pp. 157-159.

Campoy, A.M., Jimerez, A.P., Ivars, A.P., and Busquets Mataix, J.V., "Using Genetic Algorithms in Content Selection for Locking-Caches," *Proceedings of the IASTED International Symposia Applied Informatics*, 2001, pp. 271-276.

Campoy, A.M., Ivars, A.P., and Busquets Mataix, J.V., "Static Use of Locking Caches in Multitask Preemptive Real-Time Systems," *Proceedings of IEEE/IEE Real-Time Embedded Systems Workshop*, 2001.

Campoy, A.M., Ivars, A.P., and Busquets Mataix, J.V., "Dynamic Use of Locking Caches in Multitask Preemptive Real-Time Systems," *Proceedings of the 15th World Congress of the International Federation of Automatic Control*, 2002.

Campoy, A.M., Perles, A., Rodriguez, F., and Busquets Mataix, J.V., "Static Use of Locking Caches vs. Dynamic Use of Locking Caches for Real-Time Systems," *Proceeding of the IEEE CCECE Canadian Conference on Electrical and Computer Engineering*, 2003, pp. 1283-1286.

Campoy, A.M., Sáez, S., Perles, A., and Busquets Mataix, J.V., "Performance Comparison of Locking Caches Under Static and Dynamic Schedulers," *Proceeding of the 27th IFAC/IFIP/IEEE Workshop on Real-Time Programming*, 2003, pp. 129-134.

"Overcoming Barriers to the Use of Commercial Integrated Circuit Technology in Defense Systems," October 1996, http://www.acq.osd.mil/es/dut/ic/contents.htm.

Singh, Ramenderpal, "Signal Integrity Effects in Custom ICs and ASIC Design," *Wiley-IEEE* Press, November 2001.

Letter Bellon, C., Liothin, A., Sadier, S., Saucier, G., Velazco, R., Grillot, F., and Issenman, M., "Automatic Generation of Microprocessor Test Programs," *Proceedings of the 19th Conference on Design Automation Conference*, 1982, pp. 566-573.

Singh, Ramenderpal, "Signal Integrity Effects in Custom ICs and ASIC Design," *Wiley-IEEE* Press, November 2001.

Becer, Murat, et al., "Signal Integrity Management in an SoC Physical Design Flow," *ISPD*, April 2003, Monterey, CA.

Yang, Andrew and Lin, Shen, "Reshaping the SoC Power Design Flow," *EETIMES*, June 2004, http://www.eetimes.com/showArticle.jhtml?articleID=17602312.

Relyea, Claudia, "Simulators Shred the Limits of Mixed-Signal Design," *Communication Systems Design*, June 2003.

12. REFERENCES.

1. RTCA/DO-254, "Design Assurance Guidance for Airborne Electronic Hardware," April 19, 2000.

2. RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," December 1, 1992.

3. Wolfe, A., "Software-Based Cache Partitioning for Real-Time Applications," *Workshop on Responsive Computer Systems*, 1993, pp. 229-237.

4. Mueller, F., "Compiler Support for Software Based Cache Partitioning," *ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Real-Time Systems*, 1995, pp. 125-133.

5. He, Y. and Avižienis, A., "Assessment of the Applicability of COTS Microprocessors in High-Confidence Computing Systems: A Case Study," *Proceedings of the International Conference on Dependable Systems and Networks*, 2000, pp. 81-86.

6. He, Y., "An Investigation of Commercial Off-The-Shelf (COTS) Based Fault Tolerance," Ph.D. Dissertation, University of California, Los Angeles, 1999.

7. Stogdill, Ronald C., "Dealing With Obsolete Parts," *IEEE Design & Test*, Vol. 16, Issue 2, pp. 17-25, April 1999.

8. Parnell, Karen and Bryner, Roger, "Comparing and Contrasting FPGA and Microprocessor System Design and Development," White Paper, XILINX Corporation, July 2004.

9. Anghel, L., et. al., "Coupling Different Methodologies to Validate Obsolete Microprocessors," Proceedings of the Defect and Fault Tolerance in VLSI Systems, *19th IEEE International Symposium on (DFT'04)*-Volume 00 IEEE Computer Society, October 2004.

10. Parnell, Karen, "Could Microprocessor Obsolescence Be a History?" White Paper, *Xcell Journal*, Spring 2003.

11. Anghel, L., et. al., "Preliminary Validation of an Approach Dealing With Processor Obsolescence," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2003.

12. Jeffers, Robert Y., "Emerging Technologies For Electronic Parts Obsolescence Management," *6th Joint FAA/DoD/NASA Aging Aircraft Conference*, September 16-19, 2002.

13. Xilinx Inc., LavaCORE™ Configurable Java™ Processor Core, http://www.xilinx.com/products/logicore/ alliance/dsi/dsi_java_proc.pdf, April, 2001.

14. Mueller, F., "Static Cache Simulation and Its Applications," Ph.D. Dissertation, The Florida State University, 1994.

15. Colin, A. and Pauaut, I., "Worst Case Execution Time Analysis for a Processor With Branch Prediction," *The International Journal of Time-Critical Computing Systems*, 18, 2000, pp. 249-274.

16. Schneider, J. and Ferdinand, C., "Pipeline Behavior Prediction for Superscalar Processors by Abstract Interpretation," *Proceedings of the ACM SIGPLAN 1999 Workshop on Languages*, *Compilers and Tools for Embedded Systems*, 1999, pp. 35-44.

17. Ofelt, D.J., "Efficient Performance Prediction for Modern Microprocessors," Ph.D. Dissertation, Stanford University, 1999.

18. "International Technology Roadmap for Semiconductors" (ITRS), 2003.

19. "SPICE—Simulation Package With Integrated Circuit Emphasis," University of California, Berkeley, http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/

20. Thatte, S.M. and Abraham, J.A., "Test Generation for Microprocessors," *IEEE Transactions on Computers*, Vol. C-29, No. 6, 1980.

21. Klug, H.P., "Microprocessor Testing by Instruction Sequences Derived From Random Patterns," *Proceedings of 'New Frontiers in Testing', International Test Conference*, 1988, pp. 73-80.

22. Bolduc, L., "Verifying Modern Processors in Integrated Modular Avionics Systems," *10th International Symposium on Software Reliability Engineering*, 1999.

23. Letter Benso, A., Rebaudengo, M., and Reorda, M.S., "Fault Injection for Embedded Microprocessor-Based Systems," *Journal of Universal Computer Science (Special Issue on Dependability Evaluation and Validation)*, Vol. 5, No. 5, 1999, pp. 693-711.

24. McCannry, Jim, "Noise Awareness Catches Timing Flaws," *EETIMES,* October 2000, http://www.eetimes.com/story/oEG2000101650054.

APPENDIX A—PROFILE-BASED PERFORMANCE ESTIMATION

The technique of reference A-1 requires the identification of the path traces in a given program. A path is an acyclic sequence of instructions through a program. A path starts in either of the following points:

- An entry point into a procedure. The entry point could either be due to a call into this procedure or due to the return of a subroutine called by this procedure.

- The top of a backward going program arc (top of a loop).

The end of a path occurs at either of the following program points:

- An exit from a procedure. The exit is either due to a call to another subroutine or due to the return from this procedure.

- The bottom of a backward going program arc (bottom of a loop).

Once the paths are identified, the technique considers both the stall a base path has on a successor and the stall a successor imparts onto the base path as well as the overlap between paths. A dependency arc from an instruction in the base path to an instruction in a successor path is an indication of additional stall cycles for the successor path. In architectures that support out-of-order issue, instructions from a successor path may be scheduled before some instructions from the base path. Execution of instructions from both paths may also overlap in the pipeline. Typically a path has multiple successors, so the estimation technique must take into account the interactions of the base path with each successor path.

Timing estimation of a program is done using equations A-1 and A-2. These equations do not take into account the effects of branch prediction and caches (instruction, data) yet. However, the discussion in the following paragraphs explains how branch prediction and cache effects are incorporated into the timing estimation. The execution time of a program is estimated by summing the products of the execution time estimate of each path (T_i) and the number of times that path has been executed ($Count_i$). This estimation is given in equation A-1.

$$T_{program} = \sum_{i=0}^{NumBBs} T_i \cdot Count_i \tag{A-1}$$

The calculation of T_i, which is shown in equation A-2, takes into account the effects due to the successor path, and f_{ij} is the frequency of the execution of the successor path j after the base path i. The term in the parenthesis is the difference between the number of execution cycles of a successor path when scheduled with the base path (t_{ij}) and the number of execution cycles of the successor path when it executes in isolation (t_j).

$$T_i = \sum_{j=0}^{NumSuccs} f_{ij} \cdot \left(t_{ij} - t_j\right) \tag{A-2}$$

The performance of this method can be improved by analyzing only the paths that include the fraction (1-ε) of the executed instructions. This method is called path pruning. The error introduced by path pruning ($\varepsilon_{\text{actual}}$) is different than ε, but the study in reference A-1 reports the maximum difference between $\varepsilon_{\text{actual}}$ and ε to be 0.039%. Arc pruning is another technique for improving the performance of the method of this section. In arc pruning the arcs in the program flow graph are sorted by contributed instructions first. Next, only the arcs that represent the first (1-ε) fraction are analyzed. For the results reported in reference A-1 the maximum difference of $\varepsilon_{\text{actual}}$ and ε is 0.033% when arc pruning is used.

Incorporating timing effects due to branch mispredictions and cache misses requires extending equation A-1. Equation A-3 gives the generalized formula that incorporates timing effects of all these features. Analysis for T_{base} (pipeline behavior) is as discussed above. T_{branch}, $T_{icache,}$ and T_{dcache} are the timing effects due to branch mispredictions, instruction cache misses, and data cache misses respectively.

$$T_{program} = T_{base} + T_{branch} + T_{icache} + T_{dcache} \tag{A-3}$$

Equation A-3 is valid when all the timing effects are independent from each other. This condition does not hold in modern microprocessors because they overlap many events like the handling of the cache misses in the main pipeline execution. However, the results in reference A-1 show that equation A-3 is good modeling of the timing effects of branch mispredictions and instruction cache misses. Equation A-3 is not a good modeling for capturing the effects due to data cache misses. The discussion at the end of this section gives more information on this.

Equation A-4 gives a way for approximating the average timing effects T_x due to branch mispredictions and cache misses. M_x is the number of misses or mispredictions and P_x is the average penalty of a miss or misprediction.

$$T_x = M_x \cdot P_x \tag{A-4}$$

Approximating P_x requires the program to be run twice; once with worst-case behavior for type (w_x) and next with perfect behavior (N_x). As equation A-5 shows, taking the difference between the worst-case behavior and the best-case behavior performance and dividing it by the number of event type x (N_x) gives P_x. W_x, B_x and N_x are the worst case behavior performance, best case behavior performance and the number of event types.

$$P_x = \frac{W_x - B_x}{N_x} \tag{A-5}$$

M_x in equation A-4 can be replaced by $m_x{\cdot}N_x$, where m_x is the miss or misprediction rate and N_x is the number of events of type x. Combining all of these yields equation A-6. The miss or misprediction rate m_x can either be approximated using the modeling discussed in sections 7.2 and 7.3 or can be collected using profiling tools.

$$T_x = m_x \cdot (W_x - B_x) \qquad \text{(A-6)}$$

Observing that the base-case time for *x* is the same as T_{base} and combining equation A-6 with equation A-3 yields equation A-7.

$$T_{program} = (1 - m_{branch} - m_{icache} - m_{dcache}) \cdot T_{base} + m_{branch} \cdot W_{branch} + m_{icache} \cdot W_{icache} + m_{dcache} \cdot W_{dcache} \qquad \text{(A-7)}$$

According to reference A-1 worst-case behavior for the branch prediction occurs when every branch is mispredicted and worst-case data cache is when every load and store misses in the data cache. Worst-case behavior of an instruction cache is when an instruction reference to the different instruction cache line misses the cache.

The study in reference A-1 applied equation A-7 at the path level and studied the effects due to the branch prediction, instruction cache, and data cache separately. The results show that the maximum error when predicting the branch prediction and instruction cache timing effects is about 5%. The prediction for the data cache timing effects is not accurate though because the maximum error is about 40%. According to reference A-1, this behavior has several causes. The first reason is that the data cache effects are not correlated with the execution paths, but the data is collected at the path level. Second, when there is enough instruction level parallelism in the code, handling a data cache miss can be completely overlapped with other useful work. Third, the worst-case scenario that assumes that each load and store operation misses the data cache overestimates the actual worst-case. Finally, there are multiple data-dependent effects that are difficult to predict.

REFERENCES.

A-1. Ofelt, D.J., “Efficient Performance Prediction for Modern Microprocessors,” Ph.D. Dissertation, Stanford University, 1999.