Advanced Technology Series

Processor Design and Implementation for Real-Time Testing of Embedded Systems

G. Walters, E. King, R. Kessinger
CPU Tech, Pleasanton, CA 94588
R. Fryer, ATAC, Ridgecrest, CA 93555

Presented at DASC (Digital Avionics Systems Conference) '98
October 31 - November 6, 1998
Bellevue, WA

CPU TECH

CPU Tech
4900 Hopyard Rd., Suite 300
Pleasanton, CA 94588
Tel: (925) 224-9920 / Fax: (925) 227-0539
email: sales@cputech.com
web: www.cputech.com

Processor Design and Implementation for Real-Time Testing of Embedded Systems

G. Walters, E. King, R. Kessinger, CPU Tech, Pleasanton, CA 94588
R. Fryer, ATAC, Ridgecrest, CA 93555

Presented at DASC (Digital Avionics Systems Conference) '98
October 31 - November 6, 1998
Bellevue, WA

Abstract

As more complex devices with higher levels of integration are inserted into real-time systems, traditional testing methods are becoming obsolete. The most difficult obstacle to thorough testing of real time embedded systems is the lack of visibility into the operations of processing elements while application software is executing. It is now possible to design and implement processors for embedded applications that are binary compatible with commercial instruction sets and have specific features for visibility to facilitate the test, debug, and maintenance of real-time processing systems. These features include Real Time Non-intrusive Instrumentation (RTNI) and Behavioral Verification Technology™ (BVT™) and do not interfere in any way in the operation of the system.

RTNI increases developer productivity by enabling direct observation of processor operation during system development, support and maintenance. BVT is used to automatically test the correct functional behavior of the integrated hardware and software against the system specification. The combination of RTNI and BVT significantly reduces system validation time, risk and cost, while increasing the coverage and assurance level. These features can be implemented in processors that are very high performance, low power, commercial grade, or radiation-hardened. Application of this approach is underway in the development of processors for military and commercial applications.

Background

Historically, DOD-designed embedded processors contained in avionics systems have incorporated 'test hooks' to facilitate the debugging of the system and software. Specifically, the Air Force's MIL-STD-1750A family[1] and the Navy's AN/AYK-14 family[2] are two examples of processors that contain these features. The processors were typically used in the implementation of the most complex and critical functions on the aircraft.

Former Secretary of Defense William Perry's Commercial-Off-The-Shelf (COTS) directive and the demand for increased processor throughput has resulted in the insertion of high performance COTS microprocessors in complex, real-time embedded systems. Since most of the software that executes on COTS microprocessors is not real-time, and high performance real-time applications tend to be low volume, microprocessor suppliers have not included extensive test hooks, such as Real-Time Non-intrusive Instrumentation (RTNI) that support real-time software applications. Some suppliers provide performance monitoring registers[3], but these do not enable real-time operations to be traced. Other proposed approaches required an external embedded test processor[4] that can be difficult to employ if space is tight. None of the approaches solve the problem of how to non-intrusively monitor real-time what is occurring on the microprocessor chip and how to adequately test the system.

[1] J. Weber and W. Cannon, "Real-Time, Non-Intrusive Software Testing: The Technique for the Future," *SAE International Congress & Exposition*, Feb 28 1994. pp. 167-177 and JIAWG, 1989. "User Console Interface Specification," *Joint Integrated Avionics Working Group Specification J88-N1F*, R. Fryer, Ed. (Dec).

[2] Control Data Corporation. 1990. "Feasibility Study Results to Implement a Real Time Non-Intrusive Monitor into the AN/AYK-14(v) VHSIC Processor Module." NWC Contract N60530-89-C-0355. (Jan 23).

[3] *MPC750 RISC Microprocessor User's Manual*, Motorola document MPC750UM/AD, August 1997, Chapter 11, pp. 11-1 through 11-12.

[4] K. Petersén, B. Magnhagen and H. Lindén, "Embedded Test Processors (RTP)," *Baltic Electronic Conference*, 1996.

Real Time Non-intrusive Instrumentation (RTNI)

Although some benefit can be obtained by testing each software section in non-real-time, bugs often do not appear until the system has been fully integrated and is running at speed. As stated by Da Silva and Sim at GO DSP[5], "Real-time execution causes event timing and the CPU profile to change, since it is affected by real-time interrupts, real-time task-switching, and inter-task dependencies."

As microprocessors become more complex and systems on silicon becomes closer to a reality, real-time visibility into the internal operation decreases and in many cases is completely blocked. Even common features such as on-chip caches and wide buses contribute to the problem. While the microprocessor is running out of the cache, there are no bus cycles at the system bus, so a traditional bus-based debugging system is completely blind to what the microprocessor is doing. With the wide buses, the microprocessor fetches multiple instructions or operands when it does a memory access. Bus-based debugging systems are unable to locate instruction or operand boundaries within the block of memory accessed. Clearly, bus-based debugging systems are incapable of adequately controlling and debugging modern microprocessors. This forces the firmware /software developer to intrusively trap real time code.

To solve these problems, CPU Tech has incorporated RTNI into its microprocessor designs. This work is based on previous work sponsored by the Navy[6] and the Air Force[7]. RTNI moves the debugging system onto the microprocessor where it has complete visibility into what the microprocessor is doing every cycle. In order to be effective, as noted by Fryer, Naval Air Warfare Center Weapons Division, "The instrumentation mechanisms must be transparent to the behavior of the software. Transparency has typically been deemed adequate if the flow of addresses in program execution is identical and if the time relationships of all software detectable events are equivalent (i.e. interrupts, sequence of procedures run, time spent in a process, etc.)."[8] The effect, as depicted in Figure 1, is as if all of the key points were accessible through a giant multiplexer.

---

[5] G. Da Silva, E. Sim "DSP Code Development Tools: Features for Modern Application Needs," white paper.

[6] R.E. Fryer, G. Vansteenkiste, and J. Van Campenhout. "Real Time Non-Intrusive Monitoring and Embedded System Simulation," in *Special Proceedings on Embedded Systems; of the Society for Computer Simulation Summer Simulation Conference, (July 1994)* and R.E. Fryer, 1973. "The memory bus monitor," In *Proceedings of the AFIPS National Computer Conference*, V42. (New York, N.Y.) 42, 75~79.

[7] W.J. Cannon, M. T. Michael, and D. D. Beeson. 1992. "Real Time, Non-Intrusive Instrumentation of Reduced Instruction Set Computer (RISC) Microprocessors," in *Proceedings of the National Aerospace and Electronics Conference (NAECON),* (Dayton, OH). IEEE (Piscataway, NJ), 550~557.

[8] R.E. Fryer, G. Vansteenkiste, and J. Van Campenhout. "The 'Instrumentation Memory', A Measuring Feature for Simulation of Real Time Embedded Software," in *Proceedings of the 1995 Society for Computer Simulation European Multiconference , (*Prague, CR), July 1994.

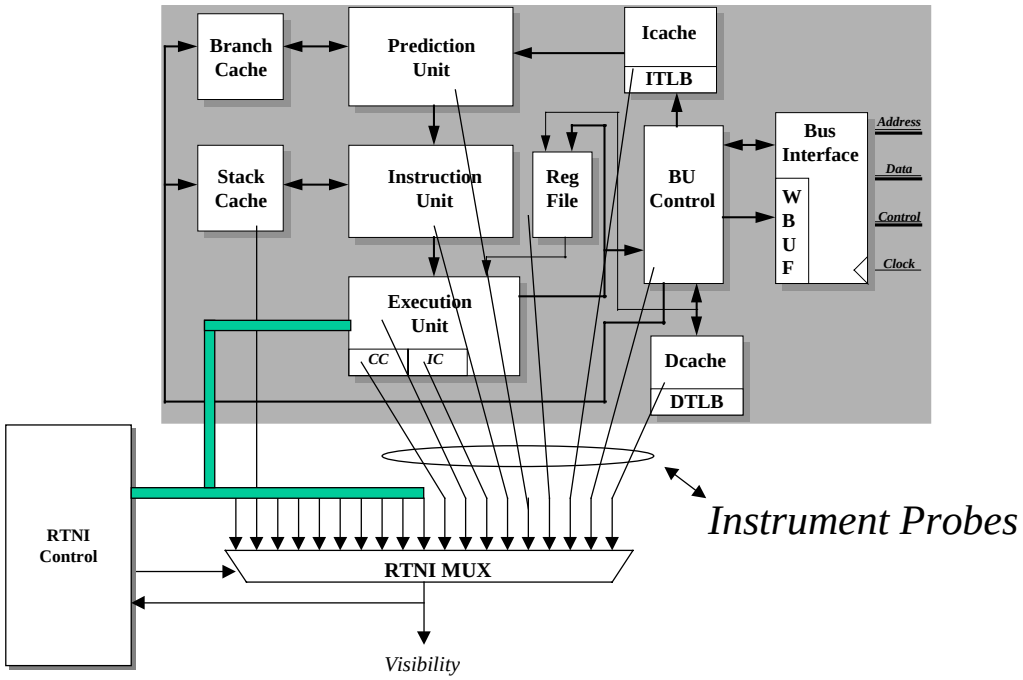

Figure 1. Representation of RTNI

External RTNI pins provides a means to transmit RTNI data off the microprocessor. In essence, parts of the traditional external debug monitor, those that control the chip's behavior, have been incorporated onto the microprocessor. To minimize the impact on the number of pins, the RTNI shares the same pins as the Built-In Self Test (BIST) and JTAG[9]. Since these functions are mutually exclusive, no conflicts arise from this arrangement.

- RTNI supports the following debug functions:
- Trace Stop/Start on Event
- Single Step
- Software/Hardware Breakpoints
- Inspect/Change Memory
- Up to (16) Hardware Traps
- Timers/Event Counters
- Event Timing
- Inspect/Change Registers
- Inspect/Change I/O Ports
- Reset/Configure Performance Monitor
- Enter/Exit Console Mode
- Reset RTNI
- Run/Stop

[9] IEEE Std. 1149.1-1990; IEEE Standard Test Access Port and Boundary-Scan Architecture.

As shown in Figure 2, the System Debug Port (SDP), external to the microprocessor, is comprised of the RTNI interface, and an Ethernet controller. The RTNI interface controls storage of RTNI information into the trace buffer and communicates RTNI control and data information with the host debugging station. Placing RTNI on chip and SDP on the microprocessor board means there are no other boards that need to be plugged into the system in order to debug it.

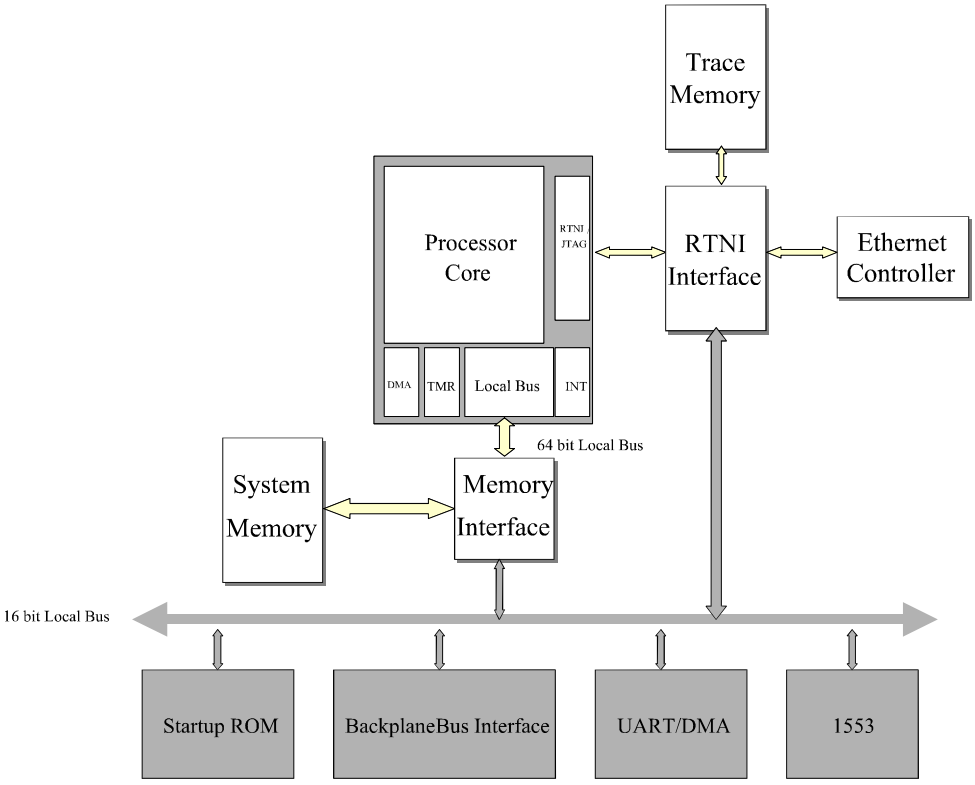

Figure 2. System Block Diagram

The RTNI interface chip is comprised of an ASIC/FPGA. The interface itself is made up of an 8-bit command bus, 32-bit data bus, and 14-bit status bus. The nineteen commands support the functions listed above. The data bus is used to:

- Specify the register number for read/write register commands
- Specify the address for read/write memory/IO commands
- Specify the data value for write commands
- Provide the data value for read commands
- Provide the trace data when tracing is enabled
- The status bus is used to indicate:
- RTNI can accept a new command
- Current RTNI command complete
- Current state of the microprocessor
- Valid trace data

- First word of multi-word trace packet
- Hit/Miss status for the internal breakpoint registers

The breakpoint registers are loaded using the 'Write Register' RTNI command. Individual breakpoints are enabled or disabled through a mask register. Once a breakpoint is encountered, it can either stop the processor execution, or start/stop tracing. The amount of trace data is limited solely by the amount of Trace Memory contained on the board. The chaining of the breakpoints is accomplished through the RTNI interface logic.

As implemented by CPU Tech, RTNI readily supports industry standard symbolic debuggers through an Application Programming Interface (API). It can also interface to simulators and other tools.

RTNI has been implemented as a general-purpose diagnostic port architecture. The architecture is open, scalable and consistent with commercial development tools, resulting in a low power, highly integrated solution for high-end embedded applications. It is incremental and compatible to the extensive R&D efforts in the industry for next generation high-end embedded systems.

Behavioral Verification Technology™ (BVT™)

While a key element, especially for software debug, RTNI is only part of the solution for testing embedded systems. RTNI provides the means to test, but BVT provides the actual diagnostics tests.

Embedded microprocessors are growing exponentially more complex with each new generation. Traditional methods of verification, which rely on manual, ad-hoc testing using existing applications software, are inadequate to guarantee that all of the functions of a microprocessor and its embedded systems are actually working properly.[10] Existing operating systems and application software do not utilize all of the functionality of these microprocessors. Worse yet, with ad-hoc methods it is unknown which functions are being tested and which are not. Future software is also at risk. Embedded systems may contain design flaws which are not exposed by today's software, but will cause catastrophic errors in future software programs.

Testing systems using exhaustive pseudo-random testing is not feasible. To perform and validate all combinations of possible instructions, modes, addresses and data patterns would take longer than the lifetime of the sun for even a simple microprocessor. The ability to properly validate all of the system's functions is the major problem faced by embedded system developers.

[10] M. Scheitrum, A. Smith, "Behavioral Validation and Its Application to Pentium Class Processors," *PCI Developers' Conference*, 1995.

BVT approaches verification through specification-based behavioral diagnostic tests. The structured diagnostics are architected to systematically test the functions of the microprocessor and the system.

BVT embodies the following characteristics:

- It is automated, checking against expected results thereby eliminating the uncertainty of ad-hoc, random testing.
- It is based on a measurable and enforceable specification. (Claiming compatibility to an operating environment is not measurable.)
- It provides comprehensive specification coverage. It tests the defined capabilities of the processor or system, not just the ones in common usage.
- It is applicable throughout the development process as well as on the finished product.
- It integrates new tests as the specification evolves or as the developer requires.
- It is an efficient diagnostic and reporting tool, which directly identifies any problems that it detects so one knows exactly what does and does not work.
- It provides repeatable and predictable test results.
- It is easy to use.

These features enable processor manufacturers to quickly identify design flaws and validate revisions to the design.

BVT consists of an extensive set of software programs. Each program is comprised of a set of self-diagnosing tests. The expected results of each test are embodied in the test program itself. These pre-engineered expected results are created in a semi-automated method which utilizes custom development tools. As the software program performs diagnostic tests on the unit under test, any discrepancy between actual and expected results are immediately reported. All specified behaviors of each function are thoroughly tested. Each test case is composed of code that sets up the initial state of the units, performs the function, and then compares the actual results against the expected results. Each test checks that proper results are stored in registers, condition codes, memory, etc. In addition, tests are performed to detect any undesired side-effects of the function.

The validation suite is organized in a fashion such that the simplest functions are tested first. The testing of these functions does not require more complicated functions to be operational. Once a function has been verified, it can then be used to verify more complex functions. This “bootstrapping” structure allows the designer to implement and debug the design in an incremental fashion, rather than requiring him or her to implement complex functions prior to testing simple operations. While BVT exercises a large number of different test cases, it is structured such that any specific test case can be isolated and exercised individually. The modularity enhances the productivity of the development team.

The basic organization of BVT is comprised of three categories: basic function tests, corner case tests, and sequence tests. An exceptional validation suite must provide comprehensive tests in each of these areas.

Basic Function Tests

Basic function tests provide coverage of every operation in all available modes. Basic tests focus on each operation or function individually. These tests verify that the fundamentals of an operation are working properly --- the proper result is calculated, registers and memory are updated correctly, and condition codes are set properly. Basic function tests also verify that unwanted side-effects do not exist.

Corner Case Tests

Corner case testing involves validating the boundary conditions and exception cases of a function. Many corner cases can be tested in an individual fashion, and this is done throughout the validation suite. Boundary conditions may involve certain operations that cause, for example, underflow or overflow in floating point operations, or that cause address calculations to cross memory boundaries. Exception cases are also tested in this section. Of particular concern is the proper prioritization of multiple exceptions that can occur in a complex system.

Sequence Tests

Individual testing of functions ensures that each function operates properly in a stand-alone manner. Yet even if all functions work correctly individually, errors could occur when multiple functions are executed sequentially or concurrently. This is due to dependencies between instructions for register values, memory contents, and flag settings. Additional tests are necessary in the suite to verify that sequences of multiple instructions interact properly.

Embedded systems have numerous functions, from simple to extremely complex, and all functions can be validated in the manner described above. The process is applicable to all functional components, including embedded controllers. BVT can be applied during the design and after the system is built, including maintaining the system after it is fielded.

Applications

The concepts embodied in RTNI and BVT are in use today. RTNI has been implemented in a MIL-STD-1750A microprocessor, called the CPU-1750A, that provides over 12 DAIS MIPS sustained at 60 MHz. The CPU-1750A is currently being inserted in an Air Force avionics system.

The RTNI is enabling the replacement of the legacy software development environment with a modern environment that has more capabilities. The effort will be completed before the end of 1999.

RTNI is also being implemented in an x86-compatible microprocessor, called the MS1, being developed in conjunction with the Navy.[11] This effort will also be completed before the end of 1999. The RTNI interface for the MS1 will be the same as for the CPU-1750A.

BVT has been applied to the testing of microprocessors and systems since 1992. It has demonstrated that it provides more test coverage than other approaches such as vector, application and/or random testing. BVT has been licensed to major microprocessor companies.

Conclusions

Processors are the most complex building blocks ever made by man. Transistor counts in processors are doubling every two to three years. Because of the fact that functional complexity can double with only an incremental increase in transistor count, the functional complexity as well as probability of design errors is growing at an exponential rate. Traditional methods of testing cannot adequately verify the correctness of these devices. While COTS microprocessors contain some additional features to support software debug, they typically are either not documented or not supported.[12]

RTNI is now available that can keep up with the single cycle execution of modern COTS processors. It can be standardized across multiple processor types, including Digital Signal Processors. The ability to visualize real-time data allows developers to isolate where and why a particular real-time bug occurs. Otherwise, it can be a frustrating and time-consuming task to debug real-time software. Bus-based debugging systems are incapable of adequately controlling and debugging modern microprocessors. RTNI re-establishes this debugging capability.

BVT, which is based on the systematic verification of the specification, is significantly more effective in finding, identifying, correcting, and re-testing bugs than verification approaches based on using only system level applications. It can be created to validate specifications for both software and hardware. BVT provides a thorough, effective solution to the problem of validating the functionality of increasingly complex processors and embedded systems.

---

[11] E. King, G. Walters, "Dual-Mode Processor Technology for Legacy System Renewal," in *Digest of Papers from the Government Microcircuit Applications Conference (GOMAC)*, (Arlington, VA), Mar 16-19, 1998, pp. 324-327.

[12] R. Fryer, "Low and Non-Intrusive Software Instrumentation, A Survey of Requirements and Methods," *IEEE 17th Digital Avionics Systems Conference*, Oct 31-Nov 6, 1998 (these proceedings).

References

Weber, J. and Cannon, W., "Real-Time, Non-Intrusive Software Testing: The Technique for the Future," *SAE International Congress & Exposition*, Feb 28 1994. pp. 167-177.

JIAWG, 1989. "User Console Interface Specification," *Joint Integrated Avionics Working Group Specification J88-N1F*, R. Fryer, Ed. (Dec).

Control Data Corporation. 1990. "Feasibility Study Results to Implement a Real Time Non-Intrusive Monitor into the AN/AYK-14(v) VHSIC Processor Module." NWC Contract N60530-89-C-0355. (Jan 23).

MPC750 RISC Microprocessor User's Manual, Motorola document MPC750UM/AD, August 1997, Chapter 11, pp. 11-1 through 11-12.

Petersén, K., B. Magnhagen, H. Lindén, "Embedded Test Processors (RTP)," *Baltic Electronic Conference*, 1996.

Da Silva, G., E. Sim "DSP Code Development Tools: Features for Modern Application Needs," white paper.

Fryer, R. E., G. Vansteenkiste, and J. Van Campenhout. "Real Time Non-Intrusive Monitoring and Embedded System Simulation," in *Special Proceedings on Embedded Systems*; of the *Society for Computer Simulation Summer Simulation Conference*, (July 1994).

Fryer, R. E., 1973. "The memory bus monitor," in *Proceedings of the AFIPS National Computer Conference*, V42. (New York, N.Y.) 42, 75~79.

Cannon, W.J., M. T. Michael, and D. D. Beeson. 1992. "Real Time, Non-Intrusive Instrumentation of Reduced Instruction Set Computer (RISC) Microprocessors," in *Proceedings of the National Aerospace and Electronics Conference* (NAECON), (Dayton, OH). IEEE Piscataway, NJ, 550~557.

Fryer, R. E., G. Vansteenkiste, and J. Van Campenhout. "The 'Instrumentation Memory', A Measuring Feature for Simulation of Real Time Embedded Software," in *Proceedings of the 1995 Society for Computer Simulation European Multiconference*, Prague, CR, July 1994.

IEEE Std. 1149.1-1990; IEEE Standard Test Access Port and Boundary-Scan Architecture.

Hetzel, W., ed. *Program Test Methods*. Englewood Cliffs, J.JK.: Prentice-Hall, 1973.

Demillo, R.A. et. al. *Software Testing and Evaluation*. Benjamin Cummings, Menlo Park, CA 1987 pp20-21,

Gupta, N. K., R. E. Seviora, "An Expert System Approach to Real Time System Debugging," *IEEE Conference on Artificial Intelligence Applications*, December 5-7 1984.

Helmbold, D. and D. Luckham. 1985. "Debugging Ada Tasking Programs." *IEEE Software,* 2 no. 2 (Mar): 47~57.

Weiss, R. 1991. "Debugging Support Goes On-Chip.", *Electronic Engineering Times* (May 20): 49~53.

"Debug tools – not an option," editorial in *Electronic Engineering Times*, April 16, 1990.

Walters, G., E. King, R. Kessinger, "Processor Design and Implementation for Real-time Testing of Embedded Systems," *IEEE 17th Digital Avionics Systems Conference*, Oct 31-Nov 6, 1998 (these proceedings).