Trace and debug port based watchdog processor

Balázs Scherer
Department of Measurement and Information Systems
Budapest University of Technology and Economics
H-1117 Budapest, Magyar Tudósok krt. 2. IE444
scherer@mit.bme.hu

Gábor Horváth
Department of Measurement and Information Systems
Budapest University of Technology and Economics
H-1117 Budapest, Magyar Tudósok krt. 2. IE444
horvath@mit.bme.hu

***Abstract*—Novel architectures and processors are designed to satisfy the needs specified by functional safety standards, like IEC 61508 and ISO 26262. An example for this devices is the TMS570 from Texas Instruments, which is a SIL3 (Safety Integrity Level 3) capable microcontroller. The question is whether a specialized hardware alone, like the TMS570 is enough for reaching safety goals or additional efforts should be done. This paper presents our novel idea of trace and debug port based watchdog processor that increase the safety for systems designed with traditional microcontrollers or with modern dual core safe ones. The approach presented is using the debug and trace hardware blocks (present in nearly every 32-bit microcontrollers introduced into the market in the last 5 years) to see inside the guarded system and to make statements of its healthiness. The paper shows the benefits and capability of such debug and trace hardware based watchdog control. We also introduce the bottlenecks of this approach and make suggestions to eliminate these, by making minor modifications to the existing trace blocks of the ARM CoreSight architecture.**

Keywords-component; embedded; trace; real-time; ARM Cortex; CoreSight; diagnostics.

I. INTRODUCTION

Many embedded electronic devices have safety critical functions. Usually a general purpose microcontroller has about more than 100 FIT (Failure-In-Time) failure rate. Therefore traditionally some kind of system level strategy is needed to reach the higher SIL levels like SIL2 or SIL3. These strategies include hardware, software, information and time redundancy, like dual processor architectures, watchdog processors, software self tests and so one. Devices with such redundancy can provide the required fail-safe operation, and in case of failure they can go into a safe state (like limp-home mode in automotive devices). 5 years ago embedded market statistics have shown a rapid change from 8-bit microcontrollers to 32-bit ones, and currently there is no question that 32-bit micros dominate the market. This change has a significant effect on the safety critical developments. In this chapter we introduce the latest trends of the microcontroller market focusing on their effects to the safety critical developments.

A. Microcontroller trends

In our work we focus on the ARM Cortex core based microcontrollers, because these devices have the leading edge in the global microcontroller market [1]. ARM has two core series which is important from our point of view. The low speed Cortex™-M core series (M0, M3, M4) for general purpose microcontrollers, and the Cortex™-R core series which is designed for real-time and safety critical systems, with optional floating point support.

B. Low-end 32-bit microcontrollers

From 2003 one of the significant trends of 32-bit microcontroller evolution was to develop devices, which are price and power consumption compatible with 8-bit microcontrollers. The flagships of these trends were the low end Cortex™-M3 core based devices, like STMicroelectronics STM32F100/101/103 series, but the breakthrough had come from the Cortex™-M0 core, which has Von Neumann architecture and therefore it is a simpler one comparing to the Harvard architecture based M3. The first Cortex™-M0 core based microcontroller series had come from NXP at 2009, and as a conclusion we can say that there are tiny 32-bit microcontrollers with up to 50MHz CPU frequency and up to 32Kbyte Flash and 8 Kbyte RAM in the below $1 price range.

C. High-end 32-bit microcontrollers with innovations for the safety critical market

Texas Instruments about 5 years ago announced a development of a microcontroller that is capable for IEC 61508 SIL3 level hardware safety without any additional redundancy. The TMS570 series is now available and it primary targets automotive and transportation market [2]. The TMS570 devices provide system-wide protection through seamless support for error detection from the processor; through the bus interconnect, to the memories. The TMS570 family integrates dual Cortex™-R4F processors in lock-step mode (F means the core is equipped with a floating point unit), working at 160MHz. The two CPUs operate cycle delayed out of sync at input and then resynchronized for output compare, which is done by the Core Compare Module. The two CPUs also have separate clock trees and they are flipped to each other to prevent physical common mode failures. The TMS570 also provides a hardware aided CPU self test and its memories are equipped with an ECC (Error Correcting Code). The TMS570 also has an Error Signaling Module (ESM) to manage the various error conditions on the TMS570 microcontroller. Any error condition can be configured to drive a dedicated device pin called ERROR to low state, which can be used as an indicator to an external monitor circuit to keep the entire system in a fail-safe state. Other semiconductor companies like

This work has been supported by the Hungarian Scientific Research Fund (OTKA), grant number TS-73496

978-1-4577-1772-7/12/$26.00 ©2012 IEEE

Freescale [3] and Renesas (probably by the influence of TMS570) have also introduced their dual safe core products.

D. Problems of safe microcontrollers

The TMS570 and other dual core safe microcontrollers provide reliable and cost effective controller platforms for modern electronics comparing to the traditional dual microcontroller based safe architectures. But these safe micros still address just a part of the problem, what was handled by the dual microprocessor based architectures. The problem not really addressed is the failures come from the software. The dual safe core is not protected against run time software failures, where the dual microcontroller architecture with different software and development platform for each controller can provide some sort of software failure protection. So this problem should be handled, because despite the software development standards and rules like AUTOSAR, MISRA-C, still there will be faults in the software. A traditional solution of this problem is some kind of watchdog, or watchdog processor.

II. TRACE, AND DEBUG PORT BASED DIAGNOSTIC

A. Software runtime failure detection

A more sophisticated approach of runtime software failure detection than a simple watchdog is the watchdog processor (WDP). A simple watchdog is many times no more than a timer or windowed timer based reset, but the watchdog processor has the ability to see inside the guarded device, and make statement about its healthiness. The watchdog processor needs some information about the inputs, outputs, and the internal state of the guarded device. The WDP approach provides high failure coverage, but it is also a costly solution. A complete I/O and communication monitoring requires much resource and a good old fashioned watchdog processor is not much less then a redundant controller.

The question is how to implement a WDP in a simpler way, which requires much less resource (I/O pins, peripheral, and therefore simpler microcontroller), than a traditional one, but still able to see inside the guarded device and make a statement about its healthiness.

A good solution could be the way that used in automotive ECU (Electronic Control Unit) grey box and HIL (Hardware in the Loop) testing. This is a diagnostic protocol based approach, where diagnostic protocols like CCP (Can Calibration Protocol), XCP (Universal Measurement and Calibration Protocol Family), KWP2000 (Keyword Protocol 2000) and UDS (Unified Diagnostic Services) are used to monitor the internal memory, and through that the state and healthiness of the ECU. This is a possible solution to the problem, but also has many drawbacks: the diagnostic channel has a very limited bandwidth, and such diagnosis cause a significant load to the main processor. Due to the drawbacks this way in not useable, but the main idea of such diagnosis, which means that the WDP checks the internal memory and state of the ECU and from these information makes the statement for healthiness of the system is good. Therefore the question is whether this memory and state checking can be done in a non-intrusive and cheap way?

B. Suggestion: Trace port based diagnostic

Modern 32-bit microcontrollers includes enhanced debug features, the ARM Cortex™ core based devices include the ARM CoreSigh on-chip trace and debug solution, where many others architectures implements the IEEE-ISTO 5001-2003 (Nexus) standard. These enhanced solutions provide non-intrusive real-time memory access without stopping the CPU. They also offer many execution trace opportunities like program, data, and ownership trace. These new debug features can provide the necessary non-intrusive diagnostic channel for the WDP. As mentioned before, we focus on ARM core based solutions. Therefore in this chapter we introduce the CoreSight architecture, and its capabilities.

C. The CoreSight on-chip trace and debug system

The CoreSight architecture [4],[5] includes many blocks, and microcontroller designers can chose to use all of them or a subset of it. The most important blocks are the following:

Debug ports: Most of the Cortex debug systems support two types of debug host interfaces. The first one is the JTAG and the second one is a new SPI (Serial Peripheral Interface) like interface called Serial-Wire (SW). The SW interface reduces the number of required signal lines to two.

AHB-AP: Advanced High-Performance Bus Access Port (AHB-AP) acts as a bus bridge to convert commands from the debug port into AHB transfers. AHB is the main bus system used in ARM core based microcontrollers. Therefore the AHB-AP allows access to the memories, private and system peripherals of the microcontroller. An external device can monitor or modify the state of the memory or peripherals of the microcontroller in a non-intrusive way without stopping the core.

The trace information in the CoreSight architecture is usually generated form three trace sources the Embedded Trace Macrocell (ETM), the ITM (Instrumentation Trace Macrocell), and the Data Watchpoint and Trace (DWT) blocks. The Trace Port Interface Unit (TPIU), formats the information form these sources into packets, and send it to an external trace capture device.

The ITM has a capability to provide a “printf” style consol messages interface to the application software. The ITM also transfers the messages of the DWT block, and what is very important that the ITM can generate timestamp packets that are inserted into a trace stream.

The DWT has a number of functionalities: among other things it can provide PC sampling at regular intervals and Interrupt events trace. The DWT also includes several counters for measuring statistical parameters like Interrupt overhead and Sleep cycles. The DWT also has the functionality to be used as a trigger of ETM. The comparators of DWT can be programmed to compare either data addresses or program counters, and on match trigger the ETM module.

The ETM block is used for providing instruction traces. To reduce the amount of data generated, the ETM does not always output exactly what addresses the processor has reached/executed. It usually outputs information about program flow and outputs full addresses only if needed. The ETM can use comparators in the DWT to generate trigger events.

The trace packets are emitted by the TPIU. The TPIU usually supports two output modes, a clocked mode, using a parallel data output port (up to 4-bit width in the case of Cortex™-M3) and a SWV (Serial Wire Viewer) mode, using single-bit UART output. SWV mode reduces the number of output signal to 1, but the maximum bandwidth for trace output is also be reduced, therefore, when instruction trace is required, the clocked mode is suggested, but for a simple data trace and/or event trace the SWV mode is usually enough. The baudrate of the SWV output can be configured in a flexible way, very similar to configuring a standard UART peripheral's baudrate.

III. SUGGESTED ARCHITECTURE

A. Suggested architecture

Chapter II had shown that, in a lowest resource combination an external WDP can monitor a microcontroller in a non-intrusive way with CoreSight on-chip trace and debug system by using only 3 pins. Two pins for an SPI like SWD communication and one pin for a one direction SWV UART communication. Chapter I had shown that there are tiny low cost (less than $1) 32-bit microcontrollers that can be used as WDP at a same cost as an 8-bit micro, but with 10 times larger performance. Therefore our suggestion is for a future safety critical system is a system with trace and debug port based watchdog processor "Fig. 1.".

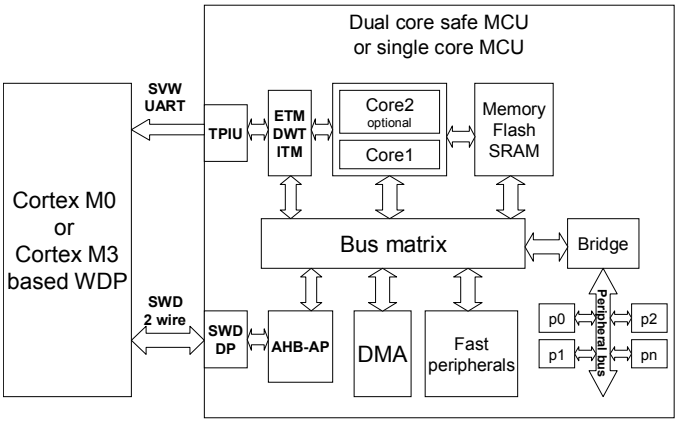


Figure 1. Suggested architecture

B. Selecting inputs for the trace and debug port based WDP

When selecting the inputs of the WDT the main question is: what type of information is needed for the WDP to make a statement about the healthiness of the system?

The healthiness of the system mainly depends on its internal state and the state of their inputs and outputs. The trace and debug port based WDP can check these states by periodically polling the memory variables regarding these properties or by making watch conditions to these variables for the trace port (limited numbers of such watch points are available). The polling based approach tends to be enough for basic state checking. This approach does not require much communication bandwidth. As an example: reading the 32-bit values of 100 variables in every 10ms (which is a normal reaction time in automotive embedded systems) require less then 1 Mbit/sec communication speed on the SWD port. Such communication speed can be easily served by a Cortex M0 core based micro running at 50 MHz.

This method is promising, but only a simplified system model can be used for checking the input, output and internal state relationships of the guarded system avoiding overrun the program memory limit of the simple WDP.

Another complementing and more universal method suggested by us is software healthiness detection. The failure model of embedded real-time software can be divided into sequential and multitasking real-time behaviour based failures. Experiences have shown that the test coverage for sequential failures is rather high. Traditional static analysis and white box tests catch these types of failures and the input, output and state polling method also provide coverage for these types of failures. The multitasking real-time failure detection is generally a harder problem. We suggest that the WDP should collect information, that helps detecting these failures, which usually appear as transient short term value changing or response time violations. There are three categories of such multitasking and real-time failures: the Timing failures, Synchronization failures and Interleaving failures [6].

These failures are typically related somehow to the embedded RTOS (Real-Time Operating System). As a starting point to what kind of failures present and should be detected it is a good idea to survey existing embedded standards related to the RTOS layer. Currently the most widespread standard related to the RTOS layer is the AUTOSAR-OS [7]. AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers, but there is a purpose to propagate the AUTOSAR to other industrial systems too. AUTOSAR-OS standard gives definitions to the terms used by operating systems developers and users. It also describes the ways of protection what should be offered by the OS. We have analysed this standard and our suggestion is to use these protection strategies, because they describe most of the possible failure in an operating system.

Many of the rules are regarding to the timing protection. An embedded real time system needs safe and accurate timing protection to ensure that Tasks and ISR-s can meet their respective deadlines. Our experiments made in a Cortex M3 based system with FreeRTOS, a commercially available RTOS have shown that the trace and debug port based WDP has the capability to track the scheduling of a guarded system by using the event tracing feature of the DWT unit. Therefore it is able to check the operation of the AUTOSAR-OS protection or provide such protection for a non AUTOSAR system.

Other protection required by the AUTOSAR OS is the Memory protection. The memory protection consists of three

parts the stack monitoring the data and code segment protection. Our experiments with the Cortex M3 and FreeRTOS based test system also has shown that the trace features are capable to monitor the stack usage of the task, and therefore signal stack faults. A code and data segment monitoring is less important, because modern fail safe microcontrollers have hardware support for such protection (like ECC).

C. *Comparing the suggested architecture to existing ones*

Out experiments has shown that the trace and debug port based WDP has the capability to monitor the internal states, inputs and outputs of the guarded device, and trace the software execution. Figure 2 compares this suggested architecture to the existing architectures.

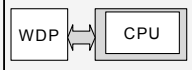
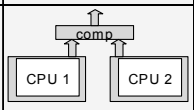
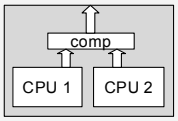
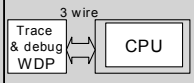
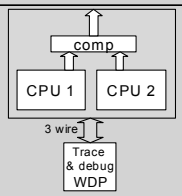
Method	diagram	Advantages	Disadvantages
Single 32 bit controller, with traditional 8/16 bit checker		relativly low cost diverse hardware	Safety depend on the capabilities of the 8/16 bit micros Diagnostic can be intrusive
Traditional dual microcontroller sollution		SIL3 reacable different hardware platform Potential for redundancy	Very costly sollution Increased complexity costly secondary micro extra board space costly and complex secondary software
Dual core safe microcontroller		SIL3 reacable reduced board space reduced software complexity less costly then the dual microcontroller	same performance as one mcu vulnerable for software failures vulnerable for comon mode failures External device for error signaling output handling
Single 32 bit microcontroller with Trace and debug port based watchdog processor		same safety as with traditional checker diverse hardware Non intrusive diagnostic	less safe then the dual microcontroller or dual core sollution.
Dual core safe microcontroller with Trace and debug port based watchdog processor		SIL3 reacable reduced board space comparing to the dual microcontroller different hardware platform less vulnerable for software failures then dual safe core less costly then the dual microcontroller	same performance as one mcu more costly then a dual core safe microcontroller WDP needs stand alone software

Figure 2. Comparing the suggested architecture to existing ones

Our experiments has shown that the trace and debug port based WDP has advantages over the traditional WDP approaches, and can complement the dual core safe micros to provide a safety level very similar to the one provided by the dual microcontroller approach, but at lower price.

Another future possible and suggested use of this concept is to integrate the Cortex M0 core based WDP to the safe microcontroller. Such integrated checker could further reduce the cost significantly. A good example of an Cortex M0™ core integration is the LPC4300 series released at 2011 by NXP (this integration has a different purpose, but demonstrate the possibility of our idea). LPC4300 series has a complex Cortex™ M4 main processor with DSP like instructions and a simple M0 core for peripheral control. It is very important, that this whole system is accessible at a $5 price range, which means adding an additional M0 core to the microcontroller system does not affect the price significantly. Therefore adding a Cortex M0 core based checker block to the TMS570 like safe microcontrollers would not modify the chip's price significantly.

IV. CONCLUSIONS

This paper has shown a novel architecture of the trace and debug port based watchdog processor. We introduced the capability of the debug and trace ports exists in modern microcontrollers and their suggested use for watchdog interface purposes. Our experiments made with our Cortex M3 based test platform have proven the concept, and on Figure 2 we presented the possible advantages of this architecture comparing to the existing ones.

However this is a promising architecture, but there are still many questions that need to be answered. The first among these is the detailed internal behaviour of the WDP. The AUTOSAR standard based analysis proved itself useful in our experiments, but further analysis and experiments needed in this field.

Our experiments also highlighted some defects of the ARM CoreSigh trace port implementation. One of these defects is that the TPIU formatting contains no information protection. There is not even a parity bit for the UART communication, which would be essential for the WDP concept. This is a significant problem and can be solved by ARM only, but solving this problem requires a very simple modification in the TPIU hardware description. Another typical problem that many microcontrollers do not have buffer for the SWV line, however the CoreSight architecture makes this possible. The lack of such buffering significantly reduces the traceable information through the SWV. Other problem with SWV is that currently not every safe core microcontroller has this interface routed to the pins of the chip.

Summary the concept of the trace and debug port based watchdog processor is promising, but many additional work and analysis needed, but these are the subject for future articles.

REFERENCES

[1] Ron Wilson's and David Blaza's Webinar, "Embedded Market Study 2011" EETimes, ESD, UBM Electronics. 2011.

[2] Karl Greb and Anthony Seely, "Design of Microcontrollers for Safety Critical Operation" ARM Techcon[5] conference presenttion, Texas Instruments 2010.

[3] Freescale white paper, "Addressing the Challenges of Functional Safety in the Automotive and Industrial Markets," 2011 Freescale Document Number: FCTNLSFTYWP.

[4] ARM Technical Reference Manual "CoreSight™ Components ",ARM DDI 0314H. 2004-2009.

[5] ARM Technical Reference Manual " CoreSight™ v1.0 Architecture Specification ",ARM IHI 0029B. 2004-2005.

[6] H. Thane. "Monitoring, testing and debugging of distributed real-time systems" *In Doctoral Thesis, Royal Institute of Technology*, KTH, S100 44 Stockholm, Sweden, May 2000. Mechatronic Laboratory, Department of Machine Design.

[7] AUTOSAR GbR. " Specification of Operating System V3.0.2 R3.0 Rev 0003", Document ID 034: AUTOSAR_SWS_OS. 2008