An RTEMS Port for the AT6981 SpaceWire-Enabled Processor : Features and Performance

Onboard Equipment and Software, Short Paper

David Paterson
STAR-Dundee Ltd.
Dundee, UK
david.paterson@star-dundee.com

David Gibson, Steve Parkes
Space Technology Centre
School of Computing
University of Dundee
Dundee, UK
davidgibson@computing.dundee.ac.uk,
sparkes@computing.dundee.ac.uk

***Abstract*— The Atmel AT6981 is a complex system-on-chip based on a SPARC LEON2-FT core, and which provides a number of peripheral devices including three multi-function SpaceWire engines and a router.**

The RTEMS real-time operating system is widely used in spacecraft systems in many roles. Its long history and open source availability make it an ideal choice for many applications. RTEMS has already been ported to many platforms, including some based on the SPARC LEON2 processor.

The process of porting RTEMS to the AT6981 is described, and the performance, both for general data processing and for SpaceWire traffic handling, is examined.

***Index Terms*— Relevant indexing terms: SpaceWire, Spacecraft Electronics, Real-Time Operating System, RTEMS.**

I. Introduction

The requirements for spacecraft on-board data handling are continually increasing in terms of demands on both processing power and network bandwidth. This has driven the development of ever more powerful and capable data processors and network controllers.

The Atmel AT6981 [1] combines a high-performance, fault-tolerant processor with multiple SpaceWire engines and a SpaceWire router, providing both data processing and network control in a single package. The inclusion of on-chip memory and a range of other peripherals and network interfaces make it a highly capable device, suitable for use in a wide range of applications.

Along with the requirements for increased processing capabilities, there is also a need for a reliable software environment to support real-time scheduling of the data handling tasks. The RTEMS operating system is an ideal candidate for this role, having proven its reliability and usefulness in use on many missions, as well as having been widely adopted in non-spaceflight applications.

Although RTEMS has been ported to LEON2-based platforms, each target system has a different configuration, so an AT6981-specific port is required in order to make full use of the device's capabilities. Porting RTEMS essentially requires the development of a target-specific Board Support Package (BSP) together with additional device drivers for the target's peripherals, and these are integrated into the RTEMS source tree in order to build the target-specific version.

II. The AT6981 System-On-Chip

Based on a SPARC V8 LEON2-FT processor running at 200 MHz, the AT6981 is ideally suited for SpaceWire-based applications with the inclusion of three powerful and flexible SpaceWire engines. Each engine contains an RMAP initiator, RMAP target and three general-purpose transmit/receive DMA channels. These SpaceWire engines are connected to a SpaceWire router which has eight external ports, providing extensive network connectivity.

Additionally, the AT6981 includes up to 1 MByte of on-chip EDAC-protected SRAM, controllers for CAN, MIL-STD-1553 and Ethernet, as well as general purpose I/O, UARTs, timers and other commonly-required interfaces.

A diagram of the AT6981 is shown below in Figure 1.

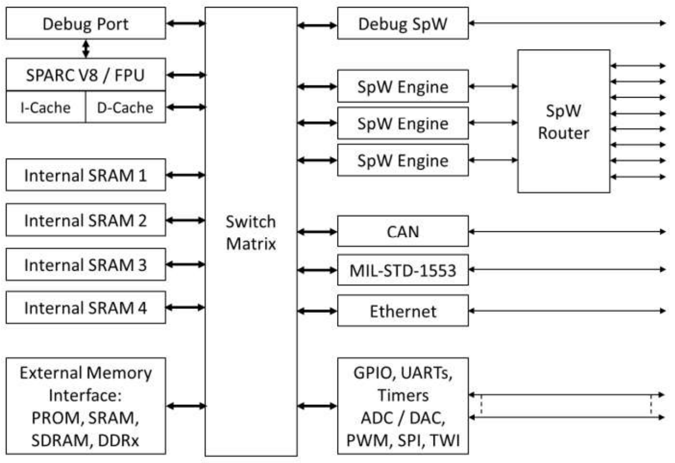

Fig. 1 – AT6981 Functional Block Diagram

The SpaceWire subsystem of the AT6981 is built around three highly capable, semi-autonomous engines which can offload much of the work involved in sending and receiving SpaceWire traffic from the main processor.

A diagram of the SpaceWire subsystem is shown below in Figure 2.

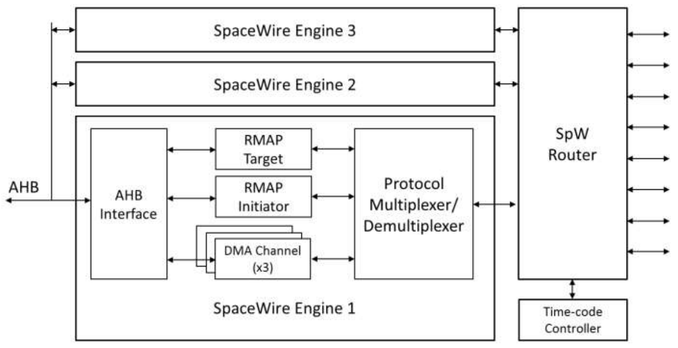

Fig. 2 – AT6981 SpaceWire Subsystem

The SpaceWire router has three internal ports and eight external ports connected through a switch matrix which allows multiple simultaneous connections between inputs and outputs.

The router is also connected to a time-code controller which indicates received time-codes, and can generate time-codes based on internal counters, hardware interrupts or on command from the processor. The time-code controller also handles and can generate distributed interrupts.

For packets being transmitted, the protocol multiplexer/ demultiplexer selects packets to be sent to the router, using a fair arbitration scheme. For received packets, the first four bytes of each packet are checked against configurable patterns and masks to determine the correct destination – RMAP target, RMAP initiator, or one of the three DMA channels.

The RMAP target accepts RMAP commands from a remote system, performs read or write operations over the AHB bus to local memory, and optionally returns a reply packet. The target supports all RMAP commands, and includes a 16 byte buffer for verified write commands.

The RMAP initiator transmits RMAP commands to read or write memory or registers on a remote system, transferring data to or from local memory via the AHB bus. The initiator is controlled by a table of transaction requests stored in memory, allowing it to transmit multiple commands and validate replies to them without processor intervention.

The three DMA channels can each transmit and receive SpaceWire packets from or to local memory. Transmitted packets can consist of one or more data chunks, allowing for separate storage of packet headers, while received packets are stored contiguously in memory. As with the RMAP initiator, transmit and receive operations are controlled by configuration tables, minimising processor overhead.

The DMA channels can also transmit and receive RMAP [2] and PUS [3] packets, using hardware CRC-8 and CRC-16 computation respectively.

With three identical SpaceWire engines, and eight external ports from the router to access the spacecraft's on-board network, the AT6981 provides a very high level of capability for data handling. The ability of these engines to operate autonomously means that this is achieved with minimal load on the main processor.

III. The RTEMS Real-Time Operating System

The RTEMS operating system has been designed specifically for use in real-time embedded environments, providing a full range of essential support features for real-time software, including mission-critical and safety-critical applications.

RTEMS has been under continuous development since the late 1980's, and has evolved over that time into a highly reliable and capable system. It has been used in a wide range of application areas, such as networking, automotive, medical, hi-fi systems, particle accelerators and, most importantly, spacecraft systems [4].

Real-time systems are differentiated from other software applications by the requirement that they must respond to events within specified time constraints – "A real-time system is one whose logical correctness is based on both the correctness of the outputs and their timeliness." [5].

Real-time requirements may be divided into two broad categories :-

- Soft real-time – in which a missed deadline does not compromise the integrity of the system or result in a catastrophic event.
- Hard real-time – in which a missed deadline causes the work performed to have no value or to result in a catastrophic event.

RTEMS is designed to handle both of these types of constraint, and implements a number of different task scheduling options to allow for flexibility in system design, and for both hard and soft real-time tasks, and variations of them, to run in the same system.

The RTEMS system is structured using a layered approach, as show in Figure 3.

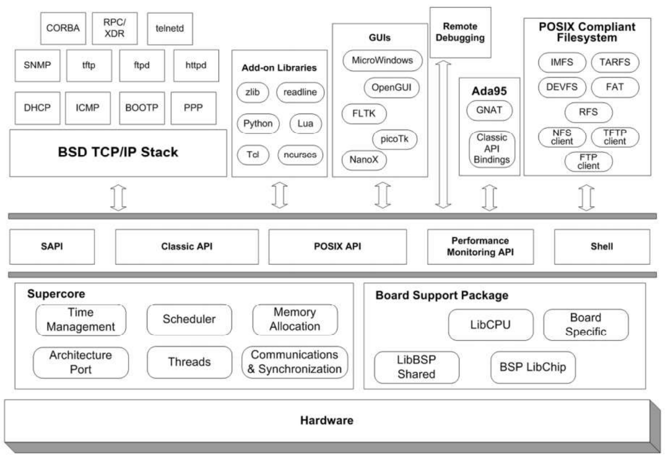

Fig. 3 – RTEMS Architecture

Although the architecture diagram shows a large number of components, the configuration mechanisms invoked when building an RTEMS system ensure that unused parts of the code base are not included in the final executable.

A number of APIs are available, including a POSIX compliant API supporting a large part of POSIX 1003.1b, such as process and thread creation and control functions and object types (semaphores, mutexes, condition variables etc.), file and directory management and memory management.

At the lowest level of the RTEMS architecture, the interface to the target hardware is managed through the Board Support Package, and this is discussed in more detail in the next section.

IV. Porting RTEMS to the AT6981

RTEMS is already in widespread use for many spaceflight applications, and is seen as a reliable and easy-to-use operating system environment for the implementation of flight software. Porting RTEMS to the AT6981 extends the range of devices which are supported, and provides a very capable hardware-software combination for many on-board data handling and communications applications.

As stated previously, RTEMS has been ported to the LEON2 in some basic configurations. However, in order to make full use of the features of the AT6981, a more complete, target-specific port was needed. This provides not only the basic integration of the processor into RTEMS, but also drivers for the built-in peripheral devices.

The first fundamental step in the process of porting RTEMS to any new target is identifying which components need to be developed, and which parts of existing ports, or parts of "standard RTEMS" can be used :-

- Does a BSP for this board exist?
- Does a BSP for a similar board exist?
- Is the board's CPU supported?

In this case, although the CPU (SPARC LEON2) is supported, no really similar board or device has yet been ported, so only some basic, shared interrupt handling code could be used. More of the common code from RTEMS, mainly related to system initialisation, could be included, but most of the BSP would have to be developed "from scratch" (albeit, based on the design and structure of other, similar BSPs).

For an initial, basic BSP, a small number of modules must be implemented :-

- Initialisation (board start-up)
- Clock driver
- Console driver
- Timer driver (optional)

The initialisation code is responsible for ensuring that the processor board is correctly initialised following a system power-on or reset. Registers and memory areas are set to known states, the stack is set up and interrupts cleared to ensure correct and reliable operation of the operating system and the application software.

The clock driver provides a reliable time reference to the RTEMS kernel, so that all primitives that require a clock tick work correctly.

The console driver, effectively a UART driver, is primarily for use in debugging and for system status report messages.

The timer driver is used by timing and benchmark tests, and although optional in the basic BSP, can be useful in determining system performance, and identifying areas which may need optimisation.

Once the basic BSP had been implemented and tested, confirming that RTEMS was operating correctly, the next step was to develop drivers for the peripheral devices on the AT6981, beginning with the SpaceWire subsystem.

In order to simplify the API for users of this feature, it was decided to write two separate device drivers, one for DMA channel management, and one for the RMAP targets and initiators. This separation also reflects the fact that these parts of each engine can operate independently.

The structure of an RTEMS device driver is relatively simple, and involves implementing a standard set of device operations – open, close, read, write and control, which map directly to the API functions typically available in most high-level language support libraries. Additionally, an initialisation function must be provided, and this is called during the RTEMS start-up sequence to carry out any device-specific initialisation which might be required.

At present, only the SpaceWire device drivers have been written, but additional drivers for other peripherals will be added in the future.

V. Performance

The testing of the RTEMS port was carried out on the STAR-Dundee AT6981 Prototype Card which contains an FPGA into which is programmed the LEON2 core, 128 KBytes of on-chip SRAM, the SpaceWire subsystem (as shown in Figure 2) and 256 MByte of DRAM. The AT6981 Prototype Card is shown in figure 4.

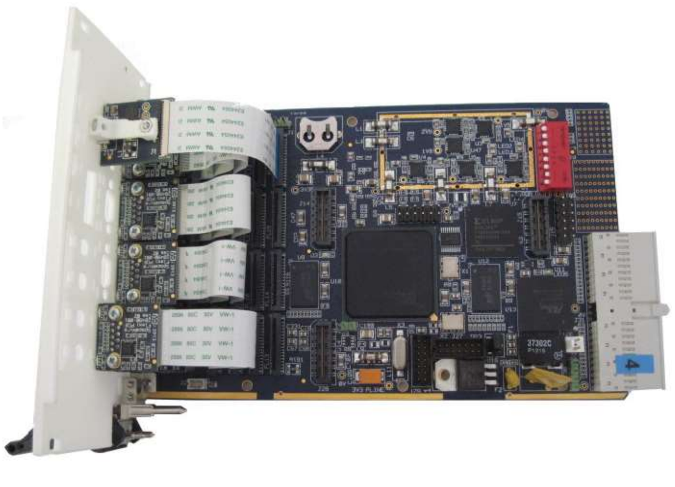

Fig.4 – STAR-Dundee AT6981 Prototype Card

The target clock speed for the production versions of the AT6981 will be 200 MHz, which should provide at least 150 MIPS Dhrystone performance, and at least 40 MFLOPS Whetstone performance. However, the Prototype card processor clock speed is limited to 30 MHz, so the measured performance is expected to be approximately one-sixth of the production device.

The SpaceWire clock on the prototype card runs at the full 200 MHz, so link speeds of up to 200 Mbit/s are supported.

Testing is still ongoing, but it should be possible to transmit and receive packets at the maximum data rate via all three SpaceWire engines simultaneously, provided they are routed through different external ports. The autonomous operation of the engines should require minimal processor overhead in handling these transactions, so processor performance is not expected to be a limiting factor in normal operation.

Final, measured performance figures will be given in the oral presentation of this paper.

VI. Conclusions

The AT6981 provides a high-performance system-on-chip solution to the ever-increasing demands for on-board data processing and network bandwidth. The flexibility and autonomous nature of the three SpaceWire engines allows for its use in a wide range of network configurations and operating modes.

RTEMS has already gained wide acceptance for use as an environment for spacecraft software, and porting RTEMS to the AT6981 extends the range of hardware which supports it. This will provide additional options to designers and developers of on-board data handling systems, providing a reliable platform on which to implement any required application software.

The open-source nature of RTEMS makes it relatively easy to configure, and to port to new target hardware. Although the AT6981 port of RTEMS currently provides only a basic BSP and drivers for the SpaceWire engines, additional drivers will be developed in the future, increasing the usability of this versatile hardware / software combination.

References

[1] "Atmel Space Rad-Hard Processors" Atmel-41005B-AEROSpaceRadHardProcessor_E_A4_052013.

[2] ECSS-E-ST-50-52C, "SpaceWire – Remote Memory Access Protocol", European Cooperation for Space Data Standardization, February 2010

[3] ECSS-E-ST-70-41A "Ground Systems and Operations – Telemetry and Telecommand Packet Utilization", European Cooperation for Space Data Standardization, January 2003

[4] "RTEMS Applications", RTEMS Wiki - http://www.rtems.org/wiki/index.php/RTEMSApplications

[5] Phillip A. Laplante, "Real-Time Systems Design and Analysis", Third Edition, John Wiley & Sons / IEEE Press, 2004