SPACEWIRE-CPCI VXWORKS SUPPORT

Session: SpaceWire test and verification

Short Paper

Iain Martin, Steve Parkes, Stuart Mills

STAR-Dundee

c/o School of Computing, University of Dundee, Dundee, DD1 4HN, Scotland, UK

E-mail: iain@star-dundee.com, steve@star-dundee.com, stuart@star-dundee.com

ABSTRACT

STAR-Dundee [1] provides both PCI and cPCI boards based on the SMCS-SpW-FPGA (Field Programmable Gate Array) device from Astrium. This FPGA is functionally representative of a radiation tolerant chip created by Astrium/Atmel [2]. These PCI and cPCI devices are therefore ideally suited to support the development and testing of on-board SpaceWire components and systems intending to use the SMCS chip. VxWorks from WindRiver [3] is a widely used real-time operating system (RTOS) in the embedded industry and is an important tool within the space industry. VxWorks driver and support software have therefore been developed for the SpaceWire-cPCI and PCI-2 boards. This paper presents the SpaceWire VxWorks driver architecture, discusses integration with Board Support Packages (BSPs) and describes the support software.

The driver performance is given by showing packet transfer rates with varying packet size and an example of a real-time application of jitter measurement using the SpaceWire-cPCI as a time code master is also presented.

1 SOFTWARE OVERVIEW

The SpaceWire VxWorks driver API is a custom interface VxWorks driver in the form of a compiled C library. Board initialisation support is provided as customisable source code with full working examples for Intel x86 and PowerPC 750 targets. Test software is provided as source code and can be used as a template to quickly develop data transfer applications. The software interface is similar to the SpaceWire PCI-2 Windows and Linux driver interfaces but it contains differences specific to an RTOS. The SpaceWire-USB RMAP [4] and Router Configuration libraries, have been ported to VxWorks and a routing table download example is shown.

1.1 SOFTWARE COMPONENTS

This section describes the software components provided to support the SpaceWire cPCI with VxWorks. The software is provided in four separate sections: Driver initialisation code which must be added to the VxWorks build, the VxWorks custom driver library, RMAP and Config support libraries and a test code suite.

Figure 1 below shows a standard development VxWorks system with the SpaceWire cPCI software integrated within a VxWorks image and a downloadable module. The initialisation code is compiled within the VxWorks image. The test code is compiled as a downloadable module and linked to the pre-compiled driver library. The RMAP and Router Configuration libraries can also be included.

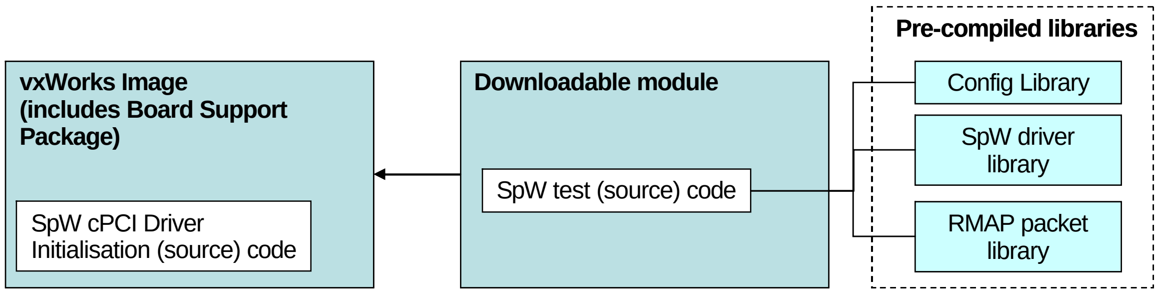

Figure 1: SpaceWire VxWorks driver software components

1.2 DRIVER INITIALISATION CODE

Driver initialisation code may need to be modified for specific targets and so is provided as source with working examples for Kontron cp620, Maxwell SCS750 and Intel x86 targets. In particular, memory mapping and the interrupt vector calculation can vary with different Board Support Packages. During initialisation, the PCI bus is scanned and an information table of detected cPCI devices is created. For each detected device this includes PCI BAR memory mapping addresses, a big/little endian flag, any error codes and an interrupt vector which is used to connect an ISR when the driver is opened. It is recommended that the initialisation code is run during VxWorks booting.

1.3 DRIVER LIBRARY

The driver library contains the VxWorks custom driver interface functions and is provided as a compiled archive. This is functionally similar to the PCI-2 Windows and Linux driver with functionality provided to support the following operations.

- Open and close devices with multiple devices supported,
- Link control which includes starting and stopping links, controlling link speed and obtaining status information,
- Data transfer functions to receive and transmit data directly to or from user buffers. Single and multiple packet transfers are supported as are transmitting and receiving raw unpacketed data. User tasks can be pended until data transfer is complete.
- Error injection and recovery,
- Time code support, and
- Low level DMA, register and memory access functions.

1.4 RMAP AND CONFIG LIBRARIES

The RMAP and Config libraries provide support for performing RMAP commands and for configuration of routers across a SpaceWire network. A router table download application is included to demonstrate the use of these libraries.

1.5 TEST CODE

The test code suite contains a range of confidence and performance test functions to demonstrate the functionality of the driver and also to provide example code which can be used as templates for data transfer applications.

2 DRIVER ARCHITECTURE

The driver architecture is tailored to the design of the SMCS3323 and PLX9056 PCI interface chips as shown in the block diagram in Figure 2. The Communication Memory Interface (COMI) of the SMCS332 transfers data to and from SpaceWire links to the dual port memory. There are two DMA channels which are used to transfer data to and from user buffers to the dual port memory. The Host Control Interface (HOCI) which bypasses the dual port memory is also supported but is significantly slower.

2.1 DATA TRANSFER

To transmit data from a user buffer out of a SpaceWire link, DMA is used to transfer data into the duel port memory. The data is then transmitted out of a SpaceWire Link using the COMI interface. Transparent internal DMA queues are used to share a single DMA channel between the three links. When a link is started, the COMI receive operation is started so any incoming data will immediately be received into the dual port memory. If a user buffer has been registered to receive data then this data will be transferred directly using DMA into the user buffer as it arrives. If no user buffer is available the data can be pended or discarded as desired. Again an internal queue is used to manage concurrent requests to DMA data from multiple links.

Each of the three SpaceWire interfaces can transmit and receive data concurrently. To enable efficient double-buffered data transfer separate internal packet transfer queues are maintained for each SpaceWire interface for both transmitting and receiving data. Each user buffer registered to transmit or receive data is allocated a buffer identifier. This can then be used to obtain the data transfer status or suspend a task until the transfer is complete or an error is detected.

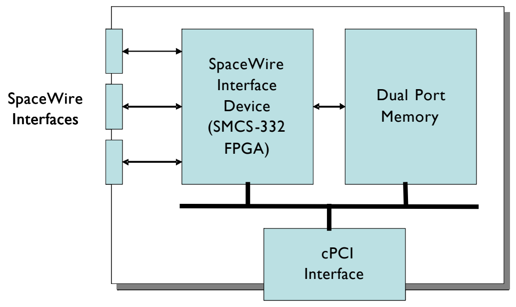

Figure 2: SpaceWire cPCI block diagram

2.2 INTERRUPT HANDLING

The driver uses a dedicated task to manage data transfer. This task acts as a "deferred procedure call" to perform any work required after an interrupt is received. The alternative model is to handle the driver interrupts directly in the Interrupt Service Routine (ISR) and so provide faster interrupt response which leads to smaller delays between packets. However the deferred procedure call model limits interrupt latency for other devices or services. The system designer can also set the priority of the driver task and so control the processor resource allocation so this is likely to be the best approach for a system with other drivers and time-critical interrupt servicing. However when servicing a high volume of small packets the task switch overhead may be an issue.

Figure 3 shows task scheduling timings during a single loopback test with packet sizes of 20,000 bytes. Task tSW0 on the left column task list is the driver task. Figure 4 shows a single loopback test with packet size of 100 bytes showing the increased processor time required to service the many extra interrupts required when using small packet sizes.

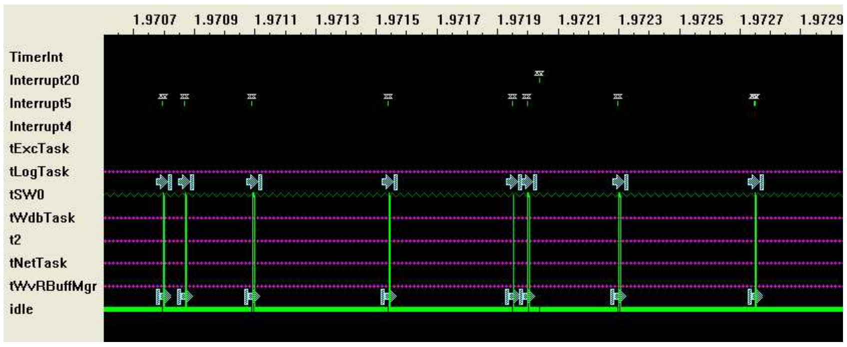

Figure 3: SpaceWire cPCI-2 driver task scheduling during a single loopback test with packet size of 20,000 bytes

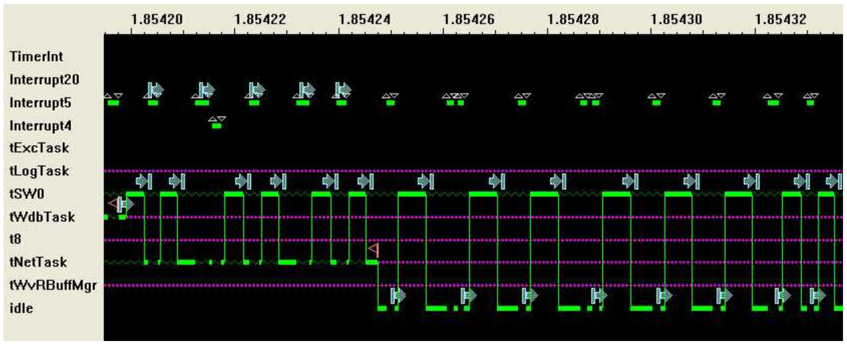

Figure 4: SpaceWire cPCI-2 driver task scheduling during a single loopback test with packet size of 100 bytes

3 PERFORMANCE

This section contains data rate transfer timings for the SpaceWire cPCI-2 board used with a Kontron cp620 bus master. Figure 5 shows the results of three data transfer tests all with varying packet size:

1. Single loopback where data is transferred out of a SpaceWire link and simultaneously read into another link connected by a SpaceWire cable. This tests two concurrent streams of data.
2. Double loop back where data is simultaneously transmitted and received out of two links connected with a SpaceWire cable. This tests four concurrent streams of data.
3. Triple loopback where data is simultaneously transmitted and received from all three links at the same time. Two links are connected with a SpaceWire cable and the other link has a single loopback cable connected. This tests six concurrent streams of SpaceWire data.

The tests use a varying packet size from 100 bytes to 5,000 bytes in steps of 100 bytes. Smaller packets sizes result in slower data rates due to increased interrupt processing overheads. When larger packets are used with multiple data streams, bus saturation is the limiting factor and is approximately 310 MBits per second when using the Kontron cp620 although this is target specific and tests with the Maxwell SCS750 gave a saturation level > 400 MBits/sec.

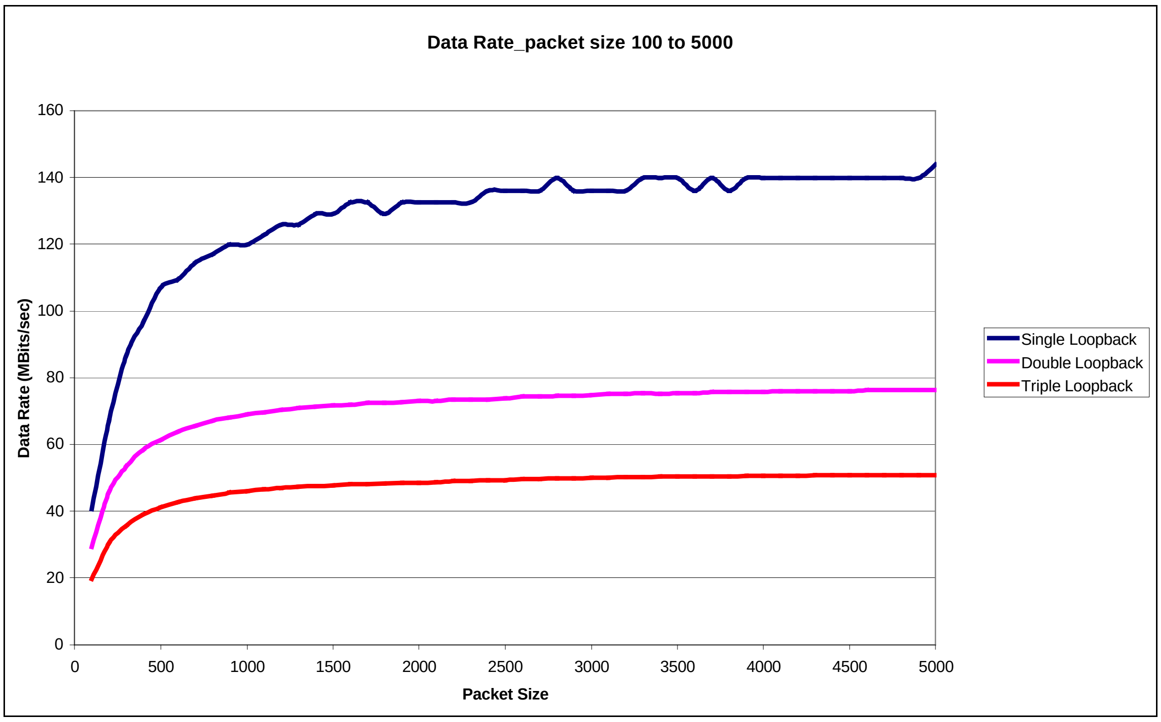

Figure 5: SpaceWire cPCI VxWorks data transfer rates with varying packet size

4 TIME CODE MASTER EXAMPLE APPLICATION

An example application was created to measure the jitter when using the cPCI board to send data packets at regular intervals controlled by time codes. A packet source application transmits a packet an exact specified time after receiving a time code. The time taken to send the packet is measured as the difference between the time code and the EOP (End-Of-Packet-marker) being detected on the SpaceWire Link. Jitter is measured as the difference between the time to send each packet and the average. The SpaceWire cPCI board was used as a packet source, time codes were provided by a SpaceWire Router-USB and the accurate timing measurements were obtained using a SpaceWire Link Analyser. The largest jitter figure measured was 1.5 microseconds.

The router is connected to the cPCI via the Link Analyser as shown in Figure 6. The router is used as a time code master and also as a packet sink to receive the packets generated by the cPCI. The Link Analyser is used to generate accurate, independent timing data. The cPCI is used to transmit packets.

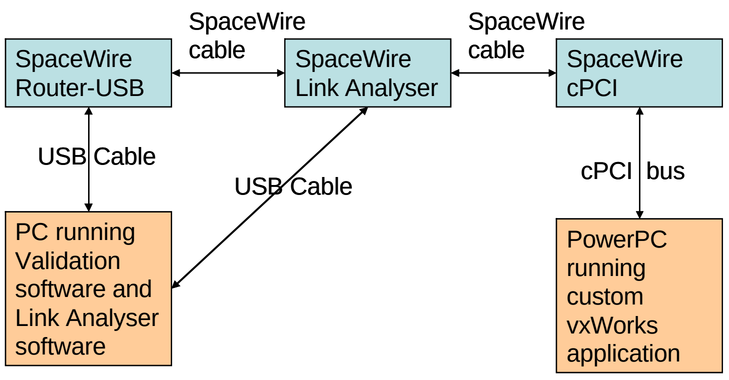

Figure 6: Jitter test setup diagram

5 CONCLUSIONS

VxWorks driver and support software has been created for the SpaceWire cPCI device. This software has a similar interface to the driver API available for the SpaceWire PCI-2 device for Windows and Linux. This allows applications using these devices to be easily ported to and from VxWorks. The RMAP and Config libraries are direct ports from the Router USB versions and so provide continuity between these products.

6 REFERENCES

1. “STAR-Dundee Web Site”, STAR-Dundee, http://www.star-dundee.com.

2. "The SMCS332SpW / SMCS116SpW SpaceWire Communication Controller ASICs", S. Fischer, L. Stopfkuchen, U. Liebstückel, P. Rastetter, L. Tunesi, EADS Astrium GmbH, 2005 MAPLD International Conference, Washington, D.C., September 7-9, 2005.

3. “WindRiver Web Site”, WindRiver, http://www.windriver.com

4. “SpaceWire Remote Memory Access Protocol”, S. Parkes, C. McClements, DASIA 2005, 30 May - 2 June, 2005, Edinburgh, Scotland. Edited by L. Ouwehand. ESA SP-602. European Space Agency, 2005. Published on CDROM., p.18.1