Multicore SoC for On-Board Payload Signal Processing

Karel H.G. Walters, S.H. Gerez, G.J.M. Smit
Computer Architecture for Embedded Systems
University of Twente
Enschede, The Netherlands
k.h.g.walters@utwente.nl

S. Baillou, G.K. Rauwerda
Recore Systems
Enschede, The Netherlands
gerard.rauwerda@recoresystems.com

R. Trautner
European Space Agency, ESTEC
Noordwijk, The Netherlands
roland.trautner@esa.int

true ***Abstract*—This paper introduces a new generic platform for onboard payload signal processing. The system is built up around an NoC with a bridge to an AMBA system which supports easy integration with existing AMBA based platforms. With the use of a pthreads interface the platform allows for simple programming and easy extension. For prototyping purposes, an implementation has been made on an FPGA together with a range of I/O options to assess its capabilities. SpaceWire and other interfaces support the extension of the demonstrator platform across multiple boards and allow to connect it to on-board networks and systems. This paper shows that novel and established chip architectures can be integrated in a way that combines their benefits, and represents a promising candidate architecture for future on-board processing platforms.**

I. Introduction

Data rates and data volumes generated by on-board instruments on science and earth observation missions are constantly increasing. This creates an increasingly high demand for power efficient on-board processing performance since the available energy is limited and the data rates available on the down links represent an unavoidable bottleneck. Existing processing platforms are usually based on a combination of ‘old’ DSP or GPP processor technology and space-qualified (but power hungry) FPGAs or dedicated ASICs [1–3].The development of specific space qualified ASICs is expensive and time consuming, in particular if they can only be used for a single mission. For this reason, new generic processing platforms need to be developed and made available for use in space. This paper describes a new platform developed in the ”Massively Parallel Processor Breadboarding” project[1]. It integrates accepted technologies with novel elements that have been developed in recent years. These new technologies include a new signal processor, a Network-on-Chip (NoC) as well as SpaceWire interfaces which are used together with an AMBA subsystem and a LEON2 processor. This hybrid architecture makes it possible to develop applications relatively fast and integrate them with existing systems while still exploiting the speed and throughput of modern technologies. It also combines the specific strengths of the individual elements, which is control-dominated processing for the LEON2, and stream based signal processing for the DSPs and the NoC.

The following sections first describe the prototype the signal processor. This is followed by a description of the different input and output interfaces, developmentenvironment and system programming, and related work. Considerations on future work including planned system improvements conclude the paper.

[1]ESA TRP contract (21986/08/NL/LvH)

II. Platform

The current implementation of the platform is mainly intended for prototyping purposes and serves as an intermediate step towards an ASIC implementation. The prototyping platform consists of two main parts, the NoC sub-system and AMBA system, as can be seen in Figure 1. The reason to include the AMBA system and not going for a full NoC-oriented system is that a lot of hardware IPs do not have an NoC interface or an interface which is easy to adapt to an NoC oriented system. Another reason is to allow easy portability of software and integration with existing systems.

Mainly the high bandwidth peripherals are connected to the NoC while the others are connected to the AMBA system. The AMBA system also provides the ability to attach the well-known LEON2 core processor to support execution of existing software with minimal changes to the source code.

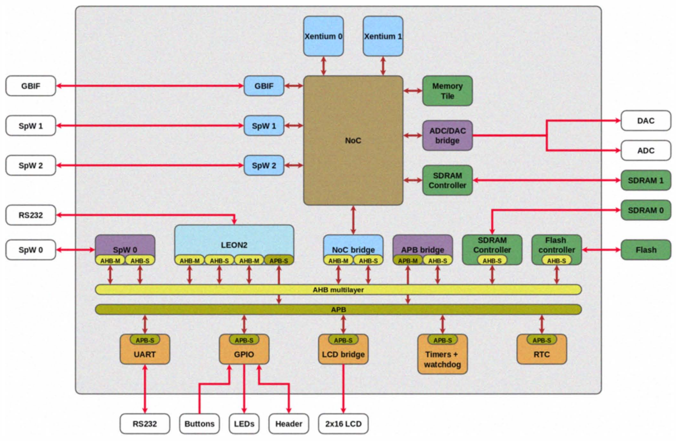

Fig. 1. Current platform architecture

A. AMBA

The AMBA system implements the AMBA-Lite standard which allows for one single master per layer. More masters on a layer would require more wires for the bus and lead to higher overall system complexity. The downside is that every master on the bus requires its own layer. So a trade-off was made between the number of layers and complexity of the system. The current implementation has four layers, one for each master on the bus, and seven slaves. The masters and slaves are indicated by AHB-M and AHB-S respectively in Figure 1.

B. NoC

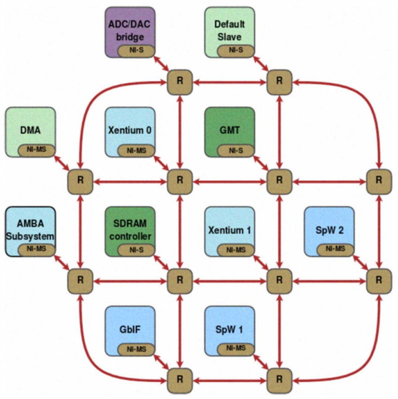

Fig. 2. NoC Architecture

The NoC is based upon earlier work of developing NoCs within our group [4] and is loosely based on the work by Bolotin et. al [5]. The NoC consists of a set of packet-switched, five-port routers. A router is connected to each neighboring router via a single 32-bit full-duplex link and to its local peripheral via four 32-bit full-duplex links. Each router schedules four different priorities. From high priority interrupt data to low priority single read and write operations. This allows for a total aggregated bandwidth of 1.6Gb/s per link at the current clock frequency of 50MHz.

All the devices on the NoC are memory mapped and accessible by devices on the AMBA and vice versa. This makes it possible for every master on the system to read and write data anywhere in the system. The LEON2 can for example read and write in local Xentium memory (see Section II-C) , as can the Xentium read and write directly to the serial port on the AMBA.

C. Xentium

The Xentium tile processor has been developed by Recore Systems. The Xentium is a programmable digital signal processing tile that is designed for baseband processing. The Xentium datapath has by default a width of 32-bits, but is customizable at design time. In this paper, we employ a default non-customized Xentium. A default implementation of the Xentium design is depicted in Figure 3.

DSP operations are performed on different processing units. One operation can be issued on each unit in each clock cycle. All operations require a single clock cycle (with the exception of load operations which require two cycles). Multiple units

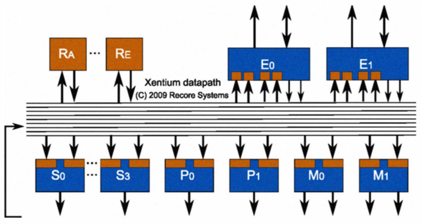

Fig. 3. Xentium core

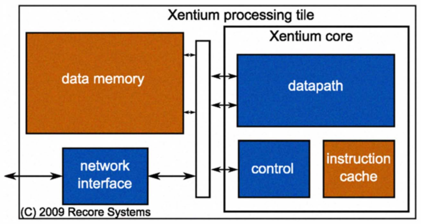

Fig. 4. Xentium together with memory and NoC interface

can be used in parallel to perform operations used in DSP kernels such as two complex multiplications per clock cycle, four multiply accumulates (MACs) per clock cycle or one radix-2 FFT butterfly per clock cycle.

The processing units in the Xentium datapath can operate in two different modes; 32-bit word or 16-bit vector mode. In vector mode, the 32-bit unit operands are interpreted as 16-bit 2-element vectors. The elements of these vectors are the low and high half-word parts of a word. Vector operations perform the same operation on the low and high parts of the vectors. Moreover, the datapath is equipped with dedicated 40-bit accumulators for improved accuracy.

The default Xentium design has two M units, four S units, two P units and two E units. The M units can perform multiply operations. The S and P units perform ALU operations (e.g. additions and subtractions) including shift and pack instructions respectively. The E units are responsible for load and store operations. Each operation can be executed conditionally.

As shown in Figure 4, the datapath is accompanied by a 32 kilobyte local SRAM, a timer, and a network interface.

III. Interfaces and Peripherals

The system has several peripherals to support the interoperability with other systems and provide options for basic debugging. The LEON2 processor was chosen as a general purpose processor. The LEON2 is used for the control-intensive parts of code and overall platform management. Although the processor has already been superseded by the LEON3 and the LEON4 [6, 7], the fully available VHDL and experience with it was the reason to choose this GPP. For a future ASIC the processor will be replaced by a fault tolerant version. The available Eclipse IDE and GRMON debugger make it easy to use and integrate with the other parts of the system.

The peripherals include an LCD display, a real-time clock, and several general purpose I/O. The real-time clock presents its values according to the CCSDS 301.0 standard for unsegmented time codes. This allows integration with CCSDS compliant telemetry packets without having to reformat the fields.

The memory tile in the NoC is an SRAM with low access times and running at the same speed as the whole platform. It is included to have a high speed generic memory to provide access to and from any of the high speed interfaces available on the platform. Especially the ADC and DAC require a place to store to and read data from without having to worry about underflow or overflow.

A. ADC and DAC

Both the analog to digital (ADC) and digital to analog (DAC) IPs (ADC/DAC bridge in Figure 1 and II-B) operate at a sample rate of up to 40 MS/s. The clock frequency of the system is the main bottleneck as there would be no room for processing if the sample rate was any higher. The AD/DA IP are customized towards an Analog Devices AD[2] and Texas Instruments DA[3]. They sample at 14-bit and 12-bit respectively and allow the samples to be packed to make efficient use of the 32-bit wide lanes on the NoC. The IP has been implemented to do all transfers via DMA such that no processing element needs to actively keep track of the transfers other than initialization.

B. SpaceWire

The system provides three SpaceWire [8, 9] connections. Two of these SpaceWire connections are directly connected to the NoC (SpW1 and SpW2 in Figure 1). This makes it possible to fully exploit the speed of the NoC and make use of low latency connections to any of the other IPs connected to the NoC. In fact, any device in the system can transfer data via the interface by accessing a memory location and configuring the DMA of the IP accordingly. The transfer clock frequency of the SpaceWire connections is 100MHz. Although the standard allows for a transfer clock of 200MHz the platform's current clock frequency limits it to 100MHz. The SpaceWire standard allows to connect any device that is compliant with the standard, which makes it relatively simple to integrate with already existing systems.

Our platform shows that SpaceWire can work well together with NoC technology in a working protoype. There is an implementation of SpaceWire for on-chip communication called SoCWIRE by Osterloh et. al. [10] but this includes a lot of overhead for the SpaceWire CODECs since it was originally intended as off-chip communication.

The SpaceWire connection on the AMBA system (SpW0 in Figure 1 has the additional RMAP target IP integrated which allows for memory mapped I/O via SpaceWire [9, 11]. Although it is only an RMAP target and not an initiator it still allows for full memory mapped I/O over the SpaceWire interface. This not only makes debugging possible but also provides a means to do data transfers on the system.

The system has memory mapped interrupt registers which provides the possibility to trigger interrupts from the SpaceWire RMAP interface. This makes it possible to handshake transfers over RMAP without having to implement an RMAP initiator IP or having to spin-lock a processing unit on a memory location.

C. Gigabit interface

The gigabit interface (GBIF in Figure 1) is implemented with the Rocket I/O available on the FPGA. The Aurora link layer protocol has been integrated with the NoC and DMA capabilities were added. The main purpose of the gigabit interface is to connect any board or other peripherals that implement the same interface to the NoC and in this way extend the platform and increase the available processing power or number of I/O interfaces (i.e. scaling up to virtual many cores).

IV. Programming

A. API

We have written an API for all the interfaces and peripherals on the system. This API makes writing clear and readable C code for the LEON2 possible. The code can be compiled with the sparc-gcc compiler and debugged with the help of the GRMON system.

The SpaceWire interface is used to supply the platform with external data. We developed a simple API available in Matlab to communicate with the SpaceWire interface of the platform over USB. This allows to initially develop an algorithm in Matlab and steadily move parts of the algorithm to the platform. When the data comes back from the platform the results can be verified using the graphical possibilities offered by Matlab.

Very efficient DSP kernels for execution on the Xentium processing tile are written in Xentium assembly. For several well known DSP kernels, efficient library functions are available. The LEON2 C program includes the Xentium assembly to be run on the platform.

B. Starting software

The bootloader of the Xentium holds execution until it receives a memory address to boot from. Typically this is an address in the RAM of the NoC, AMBA system or off-chip SDRAM. The LEON2 runs the main control thread and provides the Xentium processors with the memory locations from where to boot. The DSP kernels that run on the Xentium typically request certain variables like filter size or twiddle factor locations. These values are provided by, the LEON2 and the Xentium can reply with an interrupt when the specific kernel has completed or set a value which the LEON2 can read.

C. Pthreads

The NewLib library [12] has been ported to the SPARC architecture which makes it possible to use the pthread library. In the currently employed programming model the LEON2 starts a phtread for the DSP kernel which is running on the Xentium. The pthread on the LEON2 can run indefinitely if no change is needed to the running thread or can be switched for

[2]http://www.analog.com/static/imported-files/data_sheets/AD6644.pdf
[3]http://focus.ti.com/lit/ds/symlink/dac5662.pdf

another thread with a different DSP kernel. The LEON2 sets up a C structure containing a pointer to the DSP kernel code, the variables and the pointer to the source and target location of the data. This structure is then used in the pthread to start the Xentium. It is not needed to replicate program code for the Xentium if the Xentiums need to run the same code, they can fetch code from the same location. The only variables that would need to change are the source and target locations of the data. This approach makes it possible to switch tasks at run-time and provides a way to keep track of which task is running on which processing core.

D. Debugging

The LEON2 is implemented including the debug support unit. Via this unit it is possible to communicate with GRMON. GRMON provides a simple debug interface and a GDB server which makes it possible to communicate with any GDB capable debugger. Most commonly, the Eclipse IDE is used which can communicate directly with GDB. The debugging is eased by the fact that all the devices and peripherals are memory mapped and since the debug support unit is a master on the AHB every part of the system can be reached via the debugger. Debugging is also possible via the SpaceWire-RMAP interface which is also a master and facilitates memory mapped I/O via the SpaceWire interface.

V. Results

The hardware system is implemented on a XpressLX330T FPGA board from PLDA[4] containing a Virtex-5 LX 330T and runs at a clock frequency of 50MHz. The experience gained from the implementation and writing software for the platform are used to improve the design and build a hardware architecture which can be deployed as an ASIC.

A. Software

A number of processing kernels have been developed for the Xentium processing tile. These kernels include various sizes of complex FIR filters and FFTs. The FFTs are run with complex input samples. Table I shows some of the results obtained with 16-bit fixed point data for a single Xentium tile. These figures include the time required for the copying of the data from and to a global memory. The time it takes to copy data to the global memory is comparable to copying data to one of the interfaces. The current platform can run any two of the mentioned DSP kernels in Table I at the same time. This is possible not only because there are two Xentiums available on the platform but also because of the multiple memories and the bandwidth available on the NoC.

The two Xentiums provide a total of 400 16-bit MMACs/s, or 200 32-bit MMACs/s or 200 16-bit complex MMACs /s at the current clock frequency of 50MHz. The LEON2 delivers roughly 42 MIPS at 50MHz.

B. Synthesis

The current system runs at 50MHz on a Virtex-5 LX330T from Xilinx. The device utilization is shown in Table II.

[4]http://www.plda.com/download/doc/board/XpressLX330T/fichelx330T.pdf

TABLE I
DSP kernel speeds for a single Xentium

kernel	cycles
FIR 128	33 per sample
FIR 256	65 per sample
complex FIR 128	65 per sample
complex FIR 256	129 per sample
FFT 1024	4687 (10667 FFTs/s)
FFT 2048	11360 (4400 FFTs/s)

TABLE II
Virtex-5 LX330T Device utilization

Resource	Used	Available	Utilization
IO pins	283	960	29.48 %
Global Buffers	13	32	40.62 %
Function Generators	147977	207360	71.36 %
CLB Slices	36995	51840	71.36 %
Dffs or Latches	78087	207360	37.66 %
Block RAMs	94	324	29.01 %
DSP48Es	18	192	9.38 %

Although the FPGA development board is employed for emulation and used mainly for prototyping, verification and validation the following figures give a small insight into how the resources are split. The whole LEON2 system takes up $\sim 10\%$ of the FPGA design. A single Xentium tile including its internal memory, timer and DMA unit takes up $\sim 20\%$ of the FPGA design. The Xentium tile has been developed to for ASIC technology and as such is not optimized for FPGA technology. The whole NoC including the Xentiums, all the connecting IPs and memory tile takes up $\sim 88\%$ of the FPGA design.

VI. Related work

There are other platforms available which are comparable to this work.

The current COTS SoCs like the TI OMAP chip [13] in which the TI C64x is combined with the ARM cortex and an image processor. These kinds of systems exist in various flavors with different general purpose processors and different kind of DSPs. The main difference with our work is that we allow for a larger number of DSPs to be included in a scalable way via an NoC while still allowing older technologies to be incorporated. In the same category systems like the Tilera [14] can be found in which a lot of general purpose computing power is combined on a single chip via an interconnect. The main difference here is that we exploit the power efficiency of DSPs and we try to offload as much of the calculations to these DSPs as possible.

Then there are the more custom platforms which are currently used for on-board processing. One example is the processor module for the ExoMars rover[1]. This system is built up around a LEON2FT based chip (COLE) running at 64MHz with 512 Megabyte of EDAC protected SDRAM. An

AT697F (also based on a LEON2) is used as a co-processor which runs at a speed of 100MHz. This processor is used for autonomous rover navigation. A chip based on our architecture could deliver a lot more processing power while still having the ability to make use of the code that is being written for the LEON2FT processor and could eliminate the need for a dedicated navigation processor.

Another architecture which is comparable to our is one that is employed for airborne SAR processing which has been developed by Langemeyer et. al. [15]. Next to the fact that this platform is not intended to be used in space it uses an FPGA to extend the platform itself which makes it less scalable. The HiBRID-SoC they used is also very specific towards the application of SAR processing and it is based on a AHB system which makes a single SoC more difficult to extend. With our gigabit interface we can directly extend the NoC without the use of an intermediate device and we can extend the NoC itself easily by adding more routers and devices.

VII. Conclusion and Future work

In this paper we have shown that current technologies can be well integrated with novel ones. The scalability of the NoC part of the platform makes it possible to increase the processing power to match the requirements. The combination of several DSPs, a LEON2 and the SpaceWire interfaces interconnected by an NoC make this a good payload processing platform.

Next to the DSP kernels that have been implemented now, we are working on more complex applications such as the CCSDS wavelet based image compression standard and a software defined radio application. The 2D wavelet transformation has already been mapped on the Xentium as part of a hyperspectral image compressor [16]. With these applications we further investigate the platform's capabilities and issues that might arise which should be resolved before ASIC implementation.

In a future ASIC implementation we want to enlarge the NoC and number of Xentium processors as well as increasing the clock frequency. In an FP7 project (CRISP[5]) [17] we employ 9 Xentiums on an NoC running at 200MHz in UMC 90nm technology. For a space qualified design, we need to improve the platform radiation tolerance by using techniques like EDAC and hardening of specific architectural elements. Apart from hardware improvements, software scheduling will be a continuing area of research in our group, this includes topics as dynamically rescheduling of tasks based on the current condition of the hardware platform [18]. This will for example enable a task to migrate away from a malfunctioning tile with minimum or no downtime for the overall application.

References

[1] T. Hult, A. Petersn, B. Dean, and A. Winton, "The ExoMars Rover Vehicle OBC," in *In proceedings of DASIA 2010*, 2010.

[2] G. Estaves, P. Leconte, G. Vissio, and Leyre, "Super-Computers For Space Applications," Defence Technical Information Center, 2005.

[3] F. Arenou, C. Babusiauxand, F. Chreau, and S. Mignot, "The Gaia On-Board Scientific Data Handling," in *Proceedings of the Gaia Symposium "The Three-Dimensional Universe with Gaia" (ESA SP-576).*, 2004.

[4] P. T. Wolkotte, "Exploration within the network-on-chip paradigm," Ph.D. dissertation, University of Twente, Enschede, January 2009.

[5] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *Journal of Systems Architecture*, vol. 50, pp. 105–128, 2004.

[6] Aeroflex-Gaisler, "Leon3 multiprocessing cpu core," http://www.gaisler.com/doc/leon3_product_sheet.pdf.

[7] AeroflexGaisler, "Leon4 32-bit processor core," http://www.gaisler.com/doc/LEON4_32-bit_processor_core.pdf.

[8] European Cooperation for Space Data Standardization, "ECSS-E-ST-50-12C - SpaceWire - Links, nodes, routers and networks," http://www.ecss.nl, European Cooperation for Space Data Standardization.

[9] European Space Agency, "Spacewire web page," http://www.estec.esa.nl/tech/spacewire/.

[10] B. Osterloh, H. Michalik, and F. B, "Socwire: A spacewire inspired fault tolerant network-on-chip for reconfigurable system-on-chip designs in space applications," in *In proceedings of Spacewire conference 2008*, 2008.

[11] European Cooperation for Space Data Standardization, "ECSS-E-ST-50-52C SpaceWire - Remote memory access protocol," http://www.ecss.nl.

[12] R. J. Johnston, "Newlib," http://sourceware.org/newlib/.

[13] Texas Instruments, "Omap application processor," http://www.ti.com/omap.

[14] Tilera Corp., "Tilera," http://www.tilera.com.

[15] S. Langemeyer, C. Simon-Klar, N. Nolte, and P. Pirsch, "Architecture of a Flexible On-Board Real-Time SAR-Processor," in *In proceedings of 2005 IEEE International Geoscience and Remote Sensing Symposium, IGARSS*, 2005.

[16] K. Walters, A. Kokkeler, S. Gerez, and G. Smit, "Low-complexity hyperspectral image compression on a multi-tiled architecture," in *In proceedings of AHS2009*, jul. 2009, pp. 330 –335.

[17] H. Hurskainen, J. Raasakka, T. Ahonen, and J. Nurmi, "Multicore Software-Defined Radio Architecture for GNSS Receiver Signal Processing," *EURASIP Journal on Embedded Systems*, vol. 2009, p. 10, 2009.

[18] T. D. ter Braak, P. K. F. Hölzenspies, J. Kuper, J. L. Hurink, and G. J. M. Smit, "Run-time spatial resource management for real-time applications on heterogeneous mpsocs," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2010), Dresden.* European Design and Automation Association, March 2010, pp. 357–362.

[5]http://www.crisp-project.eu/