

# A High Performance Low Power System-on-Chip Platform Architecture

A.T. Erdogan<sup>1</sup>, T. Arslan<sup>1,2</sup> and W.-C. Lo<sup>2</sup>

<sup>1</sup>*University of Edinburgh,  
Department of Electronics & Electrical Engineering,  
Edinburgh EH9 3JL, Scotland, United Kingdom.*

<sup>2</sup>*Institute for System Level Integration, The ALBA Campus,  
Livingston, EH64 7BH, Scotland, United Kingdom.*

**Abstract:** This paper describes a System-on-Chip platform architecture for low power high performance Digital Signal Processing intensive applications. The platform is based on the AMBA SoC bus protocol and incorporates a novel interfacing scheme which utilises the bus hierarchy within AMBA in order to allow single and multiple high performance DSP Intellectual Property cores to be integrated to the SoC platform. The paper describes the overall SoC platform architecture and the integration scheme, providing results for area usage and power consumption of the main blocks in the platform with an example of a three DSP core integration case.

**Key words:** Low power, System-on-Chip, AMBA, DSP

## 1. INTRODUCTION

Today's System-On-Chip (SoC) devices are targeting complex multi-media applications where there is a need for significant amount of Digital Signal Processor (DSP) computing power in order to achieve the various data processing tasks which could include both video and audio. In order to complete these tasks various DSP Intellectual Property (IP) cores, which are optimised for the application in hand, are utilised. In order to satisfy the constraint of fast time to market effective platform architectures are required. High performance IP cores can be connected to the above platforms on a plug-in basis in order to maintain performance constraints as well as that of the fast time to market [1] [2].

Although numerous researchers in the literature have considered the development of interfaces and wrappers [3] [4], only a few discuss implementation issues and impact at the SoC level. This paper will discuss the implementation of an efficient interface strategy which utilises the bus hierarchy within the Advanced Microcontroller Bus Architecture (AMBA) [5] in order to allow single and multiple DSP IP cores to be integrated into the SoC platform utilising the LEON Processor [6]. The interface connects the IP cores directly to the AMBA bus hierarchy.

The AMBA, which defines two types of bus configurations, represents an open standardised bus architecture. The system bus, called AHB (Advanced High-performance Bus), is intended for high clock frequency system modules and enables access to high bandwidth memory devices. The peripheral bus, called APB (Advanced Peripheral Bus), is optimised for minimal power consumption and suits low bandwidth peripheral modules.

This hierarchical bus architecture is more suitable for low power consumption and high speed performance in comparison to a single bus architecture, such as standardised on-chip busses like the PI-Bus [7] or the CSI-Bus [8]. The more modules are connected to one bus the slower it operates, and consequently the latency (execution time of a transaction across the bus) of the bus increases. The total bandwidth (product of clock frequency and the byte width of the bus) request of all modules is another reason to form groups of modules around separate busses [9]. Modules, requiring high bandwidth and probably having an intensive interchange of data among each other, should be considered in the same group while modules with much lower communication needs should be bundled together. This provides the facility to power down the bus that connects the low bandwidth modules. For complex SoC designs, the increase in bus partitioning will create more than one system and peripheral bus in order to keep bus capacitances at realistic levels and to toggle these only when necessary [10].

In comparison to the bus interface design that contains a Virtual Component Interface (VCI) [11], the AMBA interface described in this paper eliminates the VCI overhead. As discussed in [12], it is less efficient to design extra VCI for both the bus interface and DSP IPs. Therefore, the AMBA interface discussed in this paper connects the DSP IPs directly to the processor without any additional control signals. This is achieved through *gated clock* and *look ahead* features. At the same time, these features have made the interface consumed less power.

The DSP IP cores used in this paper are based on an FIR filter with similar architectures [13] [14]. These are all based on a transpose direct form structure which reduces the switching activity at data inputs of the multiplier. The data input of the multiplier remains unchanged until it is multiplied by all the coefficients. In addition, the transpose direct form structure can be exploited by different numerical ordering algorithms for manipulating filter coefficients in order to reduce the switching activity at the coefficient input of the multiplier. However, the choice of the cores here are merely to demonstrate the overall integration strategy in the platform.

## 2. INTERFACING STRATEGIES

There are 2 ways of interfacing a DSP core to a processor such as LEON. These are:

1. The DSP core can be attached to the LEON processor as a co-processor. This allows high-speed communications between the LEON processor and the DSP core. According to the SPARC V8 manual, parallel processing between the DSP co-processor and LEON processor can be achieved, thus further improving the throughput and performance. However, this technique has its own disadvantages. The LEON processor can only support 1 Co-processor, thus no additional core can be added using this interface strategy. Besides, it requires its own unique Co-processor instruction

set. Furthermore, access to other peripherals in the LEON Processor requires all data to pass through the Integer Unit, thus incurring extra overheads. Therefore, the use of this interface strategy is discouraged.

2. The DSP cores can be integrated to the LEON Processor through the AMBA on-chip bus architecture. However, an AMBA Interface has to be developed for the DSP core to handle the protocols and handshakes of the AMBA specifications. Communication between the DSP cores with the Integer Unit (LEON Processor) and other peripherals are carried out over the AMBA on-chip buses as shown in Figure 1. This interface strategy offers many advantages. It allows the portability of the DSP cores. Adding new DSP cores or modifying existing cores are much easier compared to the previous option. Access to the Integer Unit and peripherals are made easy using the well-defined AMBA standards. Therefore, this interface strategy is undertaken in this project to develop an AMBA Interface.

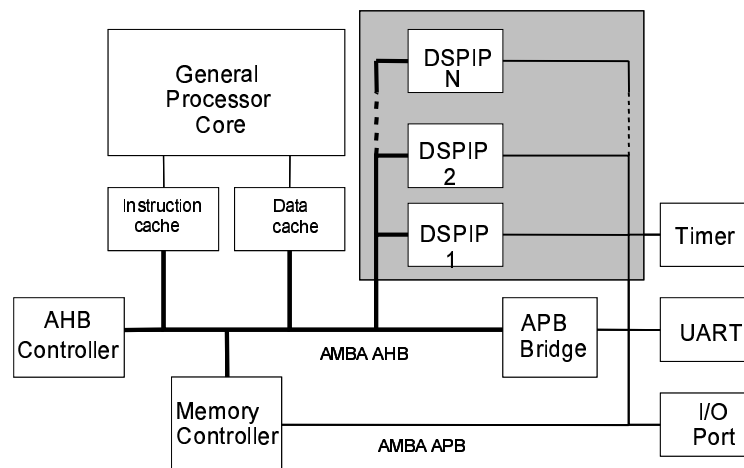


Figure 1: A Block Diagram of the SoC Platform

### 3. PLATFORM ARCHITECTURE

The overall SoC platform architecture is demonstrated in Fig. 1. The architecture is built around the LEON processor, and includes a number of high performance DSP IP cores. LEON is a synthesisable VHDL model of a 32-bit SPARC V8 compatible processor, developed by the European Space Agency (ESA) for future space missions [6]. The LEON Processor is chosen in this work because of its advance architecture. For example, the AMBA AHB and APB on-chip bus architecture allows additional modules to be added. Meanwhile, the gated clock scheme allows the reduction of power consumed and the power-down mode that allows the integer unit to be halted. All of these factors have made the LEON Processor a suitable microprocessor for this work.

The LEON processor is connected to the AMBA interface through the AHB and the APB. Within the LEON processor, the AHB is intended for high clock frequency system modules and enables access to high bandwidth memory devices, while the APB is optimised for minimal power consumption and suits low bandwidth peripheral modules. Any data that the LEON processor sends or receives (from the AMBA

interface and to the DSP IP cores) will be through these buses. The AMBA interface functions as a middleman that monitors the AHB and APB buses for any possible transaction being directed to the selected DSP IP. If a read or write operation is required, then the AMBA interface will load or store data (from the AHB or APB buses) to the DSP IP respectively. The DSP IP only accepts input data, processes this data and outputs the resulting data. Therefore, the AMBA communication protocol with the LEON processor is transparent to the DSP IP core as these protocols are handled by the AMBA interface. Likewise, the DSP IP's clock is actively controlled by the AMBA interface to save power.

#### 4. THE AMBA INTERFACE

The AMBA interface is implemented as both an AHB Slave as well as an APB device. Inputs into the interface must go through the AHB, while outputs from the interface can go through AHB or APB, depending on the urgency of output data required by the LEON Processor as shown in Fig. 2.

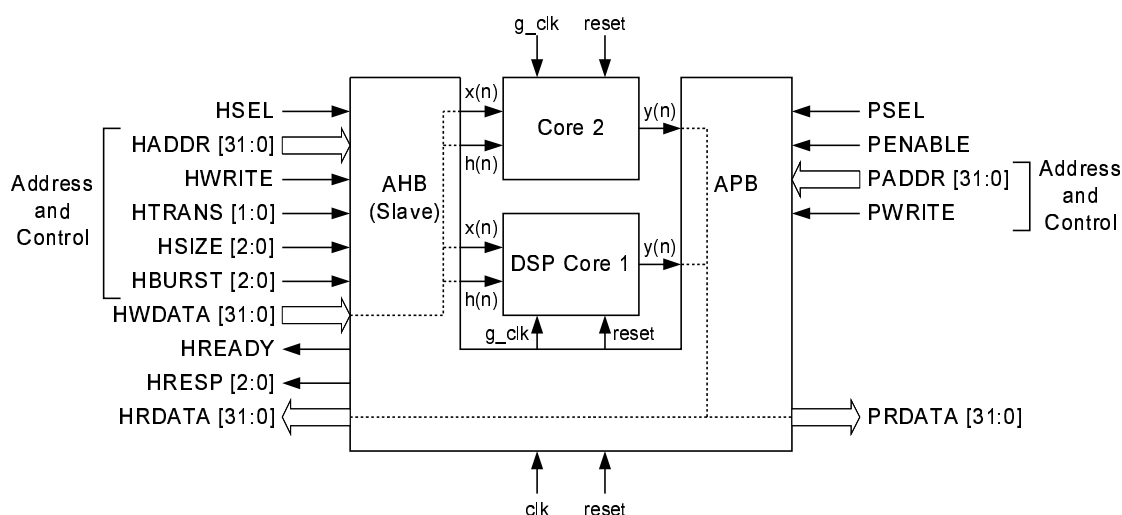


Figure 2: A Block Diagram of the AMBA Interface

The AMBA Interface is designed to be capable of supporting more than 1 core and handling different number of inputs and outputs for each core. Therefore, it is possible to connect a few cores together into one AMBA interface or allocate a separate AMBA interface for each heavy data traffic (e.g., image data) DSP cores.

Initially, the interface accepts address and control signals from the LEON Processor. The address signals are decoded to determine which core is selected, the desired input or output operation, the burst number and the PMU (Power Management Unit) mode. Meanwhile, the control signals are decoded to determine the type of transaction, type of burst operation, size of data and whether it is a read or write operation. With this information, the AMBA Interface will then route the data from the write data bus (HWDATA) to the appropriate input, or retrieve data from output and route it to the appropriate read data bus (PRDATA or HRDATA).

## 5. LOW POWER CLOCKING SCHEME

Gated clock has been employed normally to selectively stop the clock in portions of circuits where active computation is not being performed [15] [16]. In [15], the Finite State Machine (FSM) is partitioned into several sub-FSMs and these sub-FSMs are clocked when necessary. In this paper, an AMBA-based low power clocking scheme is devised for the interface to reduce the switching activity in the AMBA interface and the connecting DSP IP Cores. This clocking scheme is based on a 3 level of clocks, where the relevant module is clocked when required. The AMBA interface is first partitioned into 4 modules, as shown in Fig. 3. These are:

- (1) AHB Address and Control Decoder,
- (2) AHB Data FSM,
- (3) APB Address and Control Decoder, and
- (4) Clock Controller.

The AHB address and control decoder will decode all the address and control signals from the LEON processor and send a signal to the AHB FSM to indicate its next state operation for the next clock cycle. This is possible due to the pipelined nature of the bus where the address and data are overlapped to allow high performance operation. The decoded address also enables the decoder to decide whether to clock the FSM and the connecting cores, by sending signals to the clock controller.

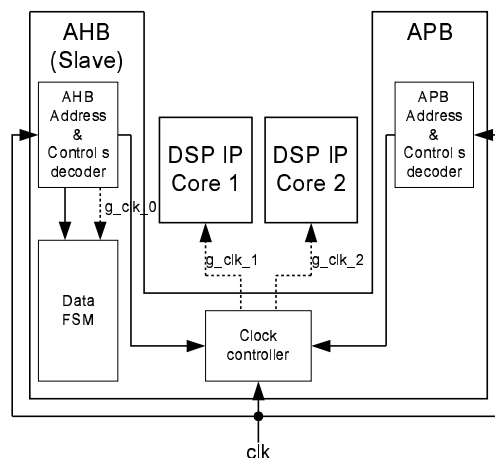


Figure 3: Partitioning of the AMBA interface for 2 DSP IPs

The AHB FSM reacts to the signals send by the AHB address and control decoder, and routes the AHB write data bus to the input of the selected DSP IP core, or routes the data from the output of the selected core back to the AHB read data bus. This AHB FSM is clocked based on the gated clock signal from the clock controller.

The APB address and control decoder decodes the address and control signals send by the APB Bridge. It will only respond to read operations while write operations are ignored. During a read operation, the output

data from the selected core will be routed back to the APB read data bus, and transferred back to the APB Bridge. Since this decoder only handles output data from the cores, a FSM is therefore not required.

The Clock controller will determine whether to clock the cores based on the signals from the AHB or APB address and control decoders. An AHB read or write operation will trigger the AHB address and control decoder to send signals to the Clock controller, indicating which DSP IP core has to be clocked. Meanwhile, an APB read operation will trigger the APB address and control decoder to send signals to the Clock controller to clock the selected core.

## 6. RESULTS AND DISCUSSION

The whole system was simulated at RTL level with Mentor Graphics' ModelSim simulator. A section of the timing diagram from the system is shown in Fig. 4. Due to the pipelined nature of the AHB operations, the data arrives on the next clock cycle following the address. Therefore, it gives the AHB Address and Control decoder 1 clock cycle time to anticipate and decide the next clock operation as well as knowing which relevant DSP IP Core to clock.

Using Fig. 4 as an example, the address of 0xA3000000 indicates that on the next clock cycle, the data of 0x000042DE has to be transfer from the AHB write data bus to the DSP IP input. The AHB address and control decoder clocks AHB FSM twice. First time is to load the FSM's next state, while the second time is to transfer the next state to current state. At the same time, the AHB or APB address and control decoder sends signal to the Clock controller to request for 1 clock cycle for the selected DSP IP core to load the data to its input.

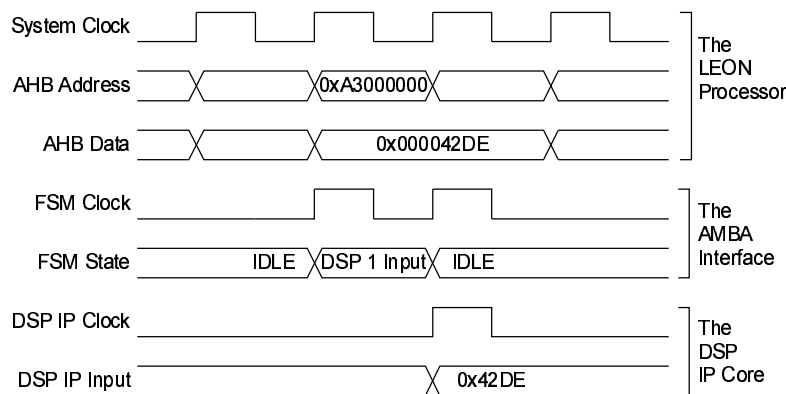


Figure 4: Timing diagram for data transfer from the LEON Processor to the DSP IP via the AMBA interface

The whole system was then synthesised to UMC 0.18 micron standard cell CMOS library, using Synopsys' Design Compiler. A maximum circuit delay of 8 ns was defined for the system. Figs. 5 and 6 illustrate the area distributions for the whole system and one of the DSP cores as an example. Clearly, most of the area is used by amba\_interface (56%) of which each DSP core contributes 18% of the area and 2% of it is used by the actual platform interface circuitry which allow the integration of the cores to the platform. This is a very

small price to pay for the added flexibility provided by the interface. The processor consumes only 26% of the whole area and the rest (18%) is used by modules such as clock generator, AHB arbiter/decoder, AHB/APB bridge, memory controller, interrupt controller, and UART. When a DSP core is examined more closely, MAC consumes most of the area (70%) followed by control (10%) and memory (7%) units.

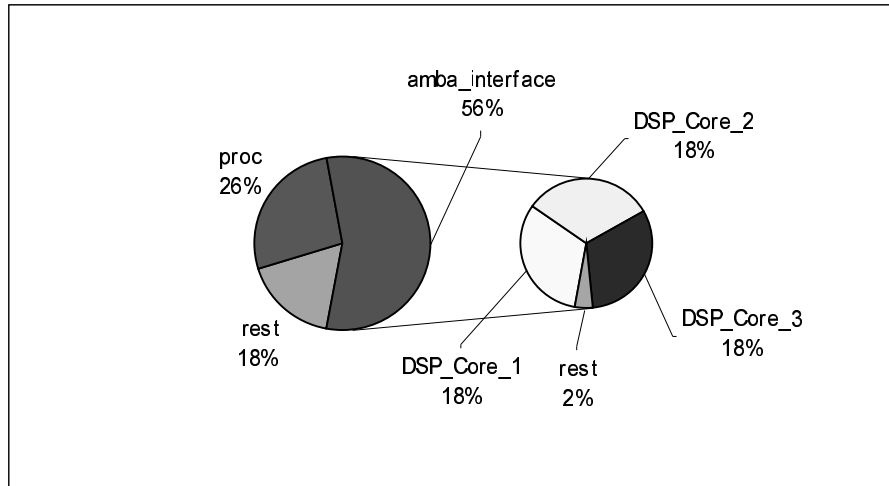


Figure 5: Area distribution of the system

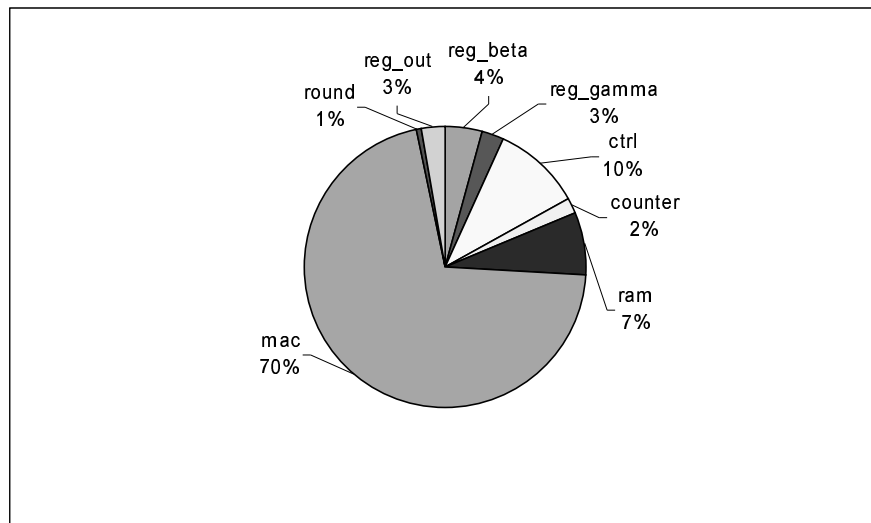


Figure 6: Area distribution for DSP\_Core\_1

The power analysis was performed by means of performing a RTL level simulation and back annotating the switching activity to the gate level netlist, using ModelSim for capturing switching activity and Synopsys's DesignPower for power computations. The power distribution of the main blocks of the system is illustrated by Fig. 7. Clearly, almost half of the power (49%) is consumed by the LEON processor, whereas *amba\_interface* consumes only 7% of the overall power, where 2% is contributed by each DSP core and 1% by the actual platform interface circuitry. As expected, the MAC unit consumes a large portion the DSP core power (66%) followed by the memory (27%) as shown in Fig. 8.

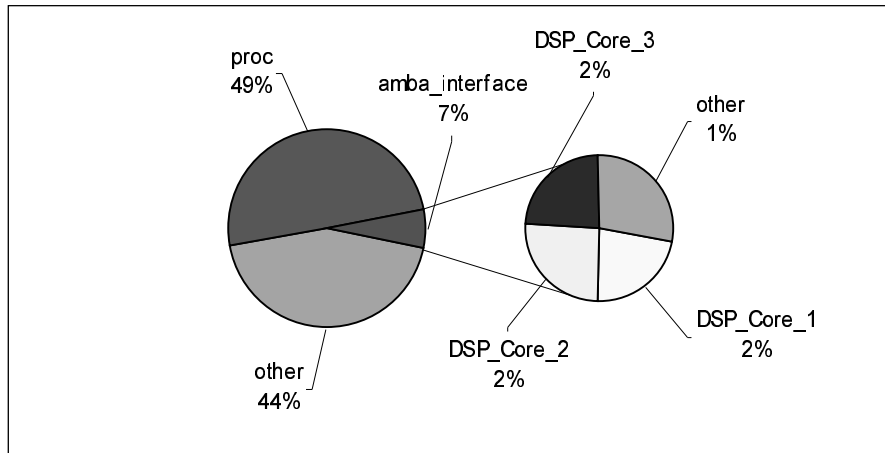


Figure 7: Power distribution of the platform

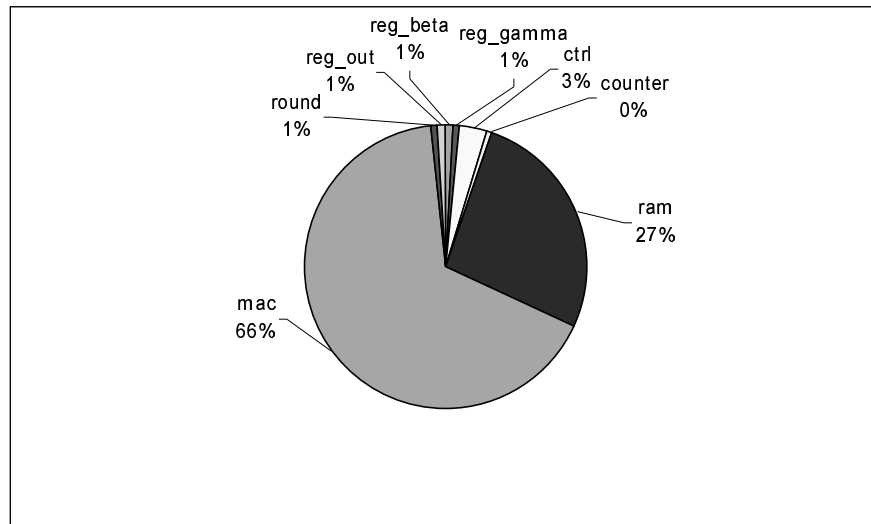


Figure 8: Power distribution for DSP\_Core\_1

## 7. CONCLUSION

The authors have presented a low power platform for the implementation of high performance DSP intensive tasks. The platform is based around the LEON processor and the AMBA bus protocol. An interfacing scheme was presented which further enhances the low power features of the AMBA architecture allowing multiple high performance DSP IP cores to be attached to the platform while maintaining low power consumption. The results have demonstrated power and area utilisation within different sections of the platform. Power and area consumption due to the interfacing scheme utilised in this work occupy a small proportion of the area when compared to the Cores and the overall platform.

## ACKNOWLEDGEMENT

The authors would like to acknowledge the support of the EPSRC under grant no GR/NO8332.

## 8. REFERENCES

- [1] HUANG, J.R., IYER, M.K., CHENG, K.T.: 'A self-test methodology for IP cores in bus-based programmable SoCs', *VLSI Test Symposium, 19th IEEE Proceedings on VTS 2001*, 2001 pp. 198 –203.
- [2] WILTON, S.J.E., SALEH, R.: 'Programmable logic IP cores in SoC design: opportunities and challenges', *Custom Integrated Circuits, 2001, IEEE Conference on 2001* pp. 63 –66.
- [3] KIM, K.W., KWANG, H.B., SHANBHAG, N., LIU, C.L., KANG S.M.: 'Coupling-driven signal encoding scheme for low-power interface design', *Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on, 2000*, pp. 318 -321.
- [4] YOO, S.J.; NICOLESCU, G., LYONNARD, D., BAGHDADI, A., JERRAYA, A.A.: 'A generic wrapper architecture for multi-processor SoC cosimulation and design', *Hardware/Software Codesign, 2001. CODES 2001. Proceedings of the Ninth International Symposium on, 2001* pp. 195 -200.
- [5] AMBA™ Specification, Revision 2.0, May 1999, © ARM Ltd., [www.arm.com/Pro+Peripherals/AMBA](http://www.arm.com/Pro+Peripherals/AMBA)
- [6] GAISLER J.: 'The LEON Processor User's Manual', Version 2.3.7, August 2001, Gaisler Research, [www.gaisler.com](http://www.gaisler.com).
- [7] PI (Peripheral Interconnect) Bus, © Open Microprocessor Initiative (OMI), [www.sussex.ac.uk/engg/research/vlsi/projects/pibus](http://www.sussex.ac.uk/engg/research/vlsi/projects/pibus).
- [8] CSI (Configurable System Interconnect) Bus, © Triscend Corp., [www.triscend.com/products/IndexTechLit.html](http://www.triscend.com/products/IndexTechLit.html).
- [9] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNeilly and L. Todd, *Surviving the SOC Revolution A Guide to Platform-Based Design*, Kluwer Academic Publishers, 1999.
- [10] P. J. Aldworth, *System-on-a-Chip Bus Architecture for Embedded Applications*, IEEE International Conference on Computer Design, Oct. 10-13, 1999, Austin, Texas, USA.
- [11] HELLMICH, H.H.H., ERDOGAN, A.T., ARSLAN, T. 'Re-Usable Low Power DSP IP embedded in an ARM based SoC Architecture', *IEE Coloquim on Intellectual Property*, Edinburgh, UK, July 2000, pp. 10/1 - 10/5.
- [12] YEUNG, C., MATTHEWS, G., MORRIS, J., HAVERINEN, A., ZAIDI, J.: 'Standard bus vs. bus wrapper: what is the best solution for future SoC integration?', *Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings, 2001* pp. 776 -776.
- [13] ERDOGAN, A.T., ARSLAN, T.: 'Low power multiplication scheme for FIR filter implementation on single multiplier CMOS DSP processors', *Electronics Letters*, Volume: 32 Issue: 21, 10 Oct. 1996, pp. 1959 -1960.
- [14] ERDOGAN, A.T., ARSLAN, T., HORROCKS, D.H.: 'Low power multiplication schemes for single multiplier CMOS based FIR digital filter implementations', *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on*, Volume: 3, 1997 pp. 1940 -1943 vol.3.
- [15] OELMANN, B., O'NILS, M.: 'Asynchronous control of low-power gated-clock finite-state-machines', *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on*, Volume: 2, pp. 915 -918.
- [16] WU, Q., PEDRAM, M., WU, X.W.: 'Clock-gating and its application to low power design of sequential circuits', *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, Volume: 47 Issue: 3, March 2000, pp. 415 -420.