The Egret Platform for Reconfigurable System on Chip

Neil W. Bergmann, John Williams
School of ITEE, The University of Queensland, Brisbane Australia
{ n.bergmann, jwilliams}@itee.uq.edu.au

Abstract

Embedded systems are an appealing application domain for reconfigurable System-on-Chip (rSoC) technology. However, rSoC design is inherently a complex task with enormous freedom in design parameters such as processor, operating system, and backplane buses. Design efficiency can potentially be improved by the use of an rSoC platform which constrains these choices, and allows new designs to leverage much of the expertise of previous designs. This paper explains and justifies the design decisions for the first version of Egret, which is an rSoC prototyping platform being developed at the University of Queensland, Australia.

1. Introduction

Embedded systems are a key enabling technology for the next generation of distributed, networked computing systems variously called pervasive computing, ubiquitous computing, invisible computing or organic computing.

Ubiquitous computing may well lead to a new “design crisis”, with new product development not being limited by technological advances, but rather by the availability of embedded systems engineers who are able to “productise” these new technological developments.

A clear response to the design crisis is re-use of software and hardware designs from one application to the next. This design re-use naturally implies the development of some standard embedded system platforms (processor choices, standard peripherals, a standard operating system, and even some standard modular circuit boards) for use within a design group.

The power of the standard platform is illustrated by the wide use of so-called Wintel (Windows + Intel) platforms in many high-end embedded platforms such as information kiosks – the platforms are generally overkill in terms of hardware speed and software flexibility, but the broad knowledge-base and availability of desktop machines as prototyping environments provides a quick and relatively risk-free path to market.

This paper describes the work at the University of Queensland to develop an embedded systems platform called Egret, based on reconfigurable System-on-Chip (rSoC) technology, and aimed at low-end embedded systems applications.

2. Reconfigurable System-on-Chip

System-on-chip (SoC) technology has evolved as the predominant circuit design methodology for custom ASICs. As FPGAs reach mega-gate size, it now becomes feasible to implement a complete microcontroller, consisting of CPU, peripherals, and a limited amount of program and data memory on a single FPGA. We call such a system a reconfigurable System-on-Chip (rSoC).

The concept on an rSoC can be extended to include systems where a hardwired CPU is incorporated on the die along with the FPGA circuitry, such as those offered by Xilinx, Altera, Atmel and Triscend [1-4]. Additionally, we extend this concept of rSoC to include those systems where external memory chips (RAM, CPU program ROM, FPGA configuration ROM, Flash) are added to the integrated CPU-plus-peripherals chip.

Lysaght [5] argues that successful use of rSoC technology will be enabled by the development and use of design platforms, in the same way that platforms have supported embedded system design.

3. Platform Specifications

The primary objectives for our platform design are to further our research into reconfigurable system-on-chip for embedded and real-time systems, and to provide a platform that students can use to rapidly prototype new reconfigurable, embedded computing applications.

Secondary objectives for our platform design are to provide a straightforward path to commercialisation of prototyped designs, and to encourage collaboration with other research and commercial developers worldwide.

Based on these objectives, we have devised the following platform specifications.

Modularity: In the case of Egret, we desire modularity in three domains. Logical Hardware Modularity means the required hardware functions for the embedded system can be readily assembled from available modules, either in the form of FPGA-based IP blocks, or in the form of specific special-purpose ICs. Physical Hardware Modularity means circuit board modules can be plugged together to meet the particular interfacing, memory, transducer, networking and power supply needs of the system. Software Modularity means software modules can be added to meet device driver, networking, data management and application specific code requirements of the system.

Flexibility and Extensibility: We require that the hardware and software design of the platform should be easily extensible to handle systems which require different amounts of memory, different amount of processing power, different networking options, and different external signal interfacing.

Plug and Play: Modules should be able to be connected together in such a way that the addition of a physical hardware module should also instantiate the appropriate FPGA-based interface IP blocks, and the appropriate software drivers.

Vendor Independence: The platform should not mandate the choice of a single vendor's FPGAs, although initially the first instantiation of the platform is likely to be for one particular vendor.

Simple Design Tool Chain: We require that our platform support a simple design tool chain, so that simple real-time embedded system designs can be accomplished on the platform without very extensive lead-times.

Reconfigurability: As far as is practical, the platform should take advantage of the design flexibility offered by the use of a central reconfigurable system-on-chip. The platform must be supportive of future research endeavours such as dynamic run-time reconfiguration, self-reconfiguration and other advanced topics.

Research Support: The platform is not primarily a platform for prototyping commercial designs. Instead, it needs to be able to support our current and planned research projects, as described later in Section 5.

Ease of Manufacture: A final minimal PCB netlist should be easily derived from the prototype system, and the physical chips present on the final design should be similar, and preferably identical, to those on the modular prototype design. Modules must therefore generally be fine grained (just a few chips each), and single-purpose.

4. Platform Design

4.1. Processor Choice

The space of potential processors for an rSoC platform, with a processor embedded on the FPGA includes the following: PowerPC405 (Xilinx Virtex Pro), Microblaze softcore (Xilinx FPGAs), ARM922 (Altera Excalibur), Nios softcore (Altera), ARM7 (Triscend), 8051 (Triscend) AVR (Atmel), and third party softcores.

To encourage research into specialised processor architectures for real-time systems, and research into multi-processor rSoC architectures, our initial preference is for a soft-core processor. Specifically, we are using the Microblaze processor in our first version of Egret.

4.2. Operating System

The major choice here was to use a small real-time kernel, or to use a full-functional operating system with real-time support.

In order to leverage the availability of a wide range of Unix-based device drivers and software applications, our operating system choice is to use a version of Unix suitable for embedded applications on the Microblaze. More specifically, our first operating system will be an embedded version of Linux for processors (such as Microblaze) without an MMU, called uClinux. We have completed the porting of the operating system kernel to Microblaze.

4.3. Physical and Logical Structure

The Egret platform will consist of a modular set of PCB building blocks that can be assembled into a complete working system. Boards will be connected together with stack-through connectors, similar in principle to the PC104 form factor.

Each board has four connectors, arranged in a square around the edge of the board. The connector structure is symmetrical, permitting boards to be plugged under any of four 90 degree rotations. We are still evaluating the optimal size of these connectors – somewhere between 32 and 120 pins per edge.

A system will consist of at least one core module containing the system FPGA with the controlling CPU, and additional peripheral modules. The core module FPGA is connected to all of the data and control pins on all of the connectors. Individual peripheral cards generally restrict their connections to one "active" connector edge, while signals on other edges are merely passed through via the connector (see Figure 1).

Across the platform are three broad classes of signals:

- Global Special Purpose (GSP) signals are distributed globally to all modules in a system, with a predefined purpose. These include power, test (e.g. JTAG), a global communications protocol (such as I^2C), and a global system clock.
- Module Special Purpose (MSP) signals are dedicated resources available to each module. For example, modules may have access to two pins that are connected to dedicated clock buffer circuitry on the core FPGA, permitting local clocks to be defined on a module by module basis; and
- Finally, Module General Purpose (MGP) signals are generic user IO from the FPGA that are available to each module to be used as required.

More than four peripherals can be added, provided that there are not pin conflicts. We do not expect many systems to have more than four peripherals, and we expect many peripheral cards to have fairly low interconnection width, such as a single SPI connection.

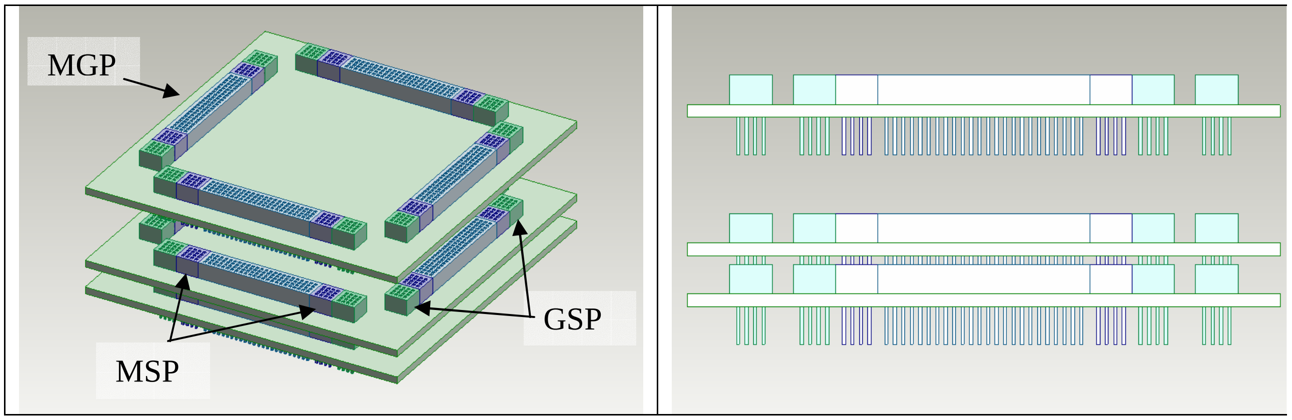

Figure 1. Visualisation of Egret's rotationally symmetrical stack-through module architecture. Global Special Purpose (GSP), Module Special Purpose (MSP) and Module General Purpose (MGP) regions of the connectors are identified.

One simple expansion approach is the use of a "gender-bending" connector board would permit modules to be flipped, resulting in a total of 8 permissible orientations. Experience with the first version of Egret will provide further insights into the advantages and pitfalls of this flexible stacking approach.

In general, as much of the digital logic as practicable will be pushed onto the core FPGA. Typical peripherals, such as a serial port or Ethernet connection, would consist of a Media Access Controller (MAC) which implements the data protocol, and a Physical device interface (PHY), which produces the correct voltage and current transformations for a particular standard. We would typically expect the MAC to be implemented as an IP block on the FPGA, and just the PHY to be on the peripheral board.

Figure 2 shows a typical modular embedded system, where modules are interconnected by an external system bus which appears on the card connectors.

In Egret, we wish to avoid a standardised off-chip logical bus architecture to which peripheral modules must conform. Rather, the system bus (or other interconnection network) is pushed back onto the core FPGA, and each external peripheral chip communicates directly to its own controller on the FPGA. Controllers communicate with one or more CPUs using the internal FPGA system interconnection network, which can be customised for individual applications. This new logical structure is shown in Figure 3.

One attribute of this approach is to confine high-speed bus clocks, address and data lines within the chip, and only "export" the necessary, and probably low-bandwidth digital signals through the connector structure and out to the applicable module. This simplifies module design for Electromagnetic Compliance (EMC) and Signal Integrity (SI) by reducing the high-frequency spectral content of external signal lines.

Once cards have been connected together into a physical Egret stack, it is necessary to instantiate the required peripheral interface cores on the FPGA, including customisation of the particular FPGA pins that the core needs to connect to. Additionally, peripheral

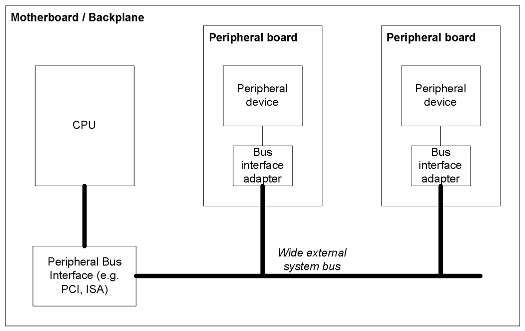

Figure 2. Typical bus-based system interconnection structure

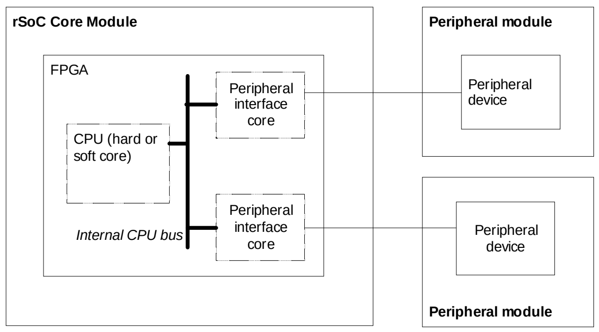

Figure 3. Egret system interconnection structure with thin peripheral interfaces

interface registers need to be mapped into appropriate CPU addresses, and appropriate device drivers loaded into the operating system.

This process of system configuration will initially be a manual process, but it should be a relatively simple process of running through a standard system design flow. Once configured, standard software development tools can be used to build applications. The aim is to specify the system at a very high level – the choice of core and peripheral modules, and their physical orientation – and have the design tool automatically assign FPGA pins to signals, detect conflicts, configure operating system drivers, and so on.

5. Research Issues

A key requirement of our Egret platform is that it should be able to expose new research questions, and provide a convenient platform for the exploration of answers to these questions. The following sub-sections describe some of our current and future research directions around Egret.

rSoC Communication Structures: rSoC products from vendors typically consist of a CPU design (soft or hardcore), a system bus interconnection structure, and a set of peripherals compatible with that system bus. Such systems provide little design flexibility. A parallel system bus is not necessarily the most appropriate interconnection structure – schemes such as serial buses, network-on-chip, or packet-switched networks may be more appropriate. In a companion paper, Andy Lee [6] explains our work in investigating alternative interconnection schemes that could be used on Egret.

Portable IP Blocks: New interconnection schemes, such as those proposed above, are most useful if existing IP blocks can be re-used with this new scheme. Tim Lee [7] is investigating an interface adaption logic scheme, which adds an appropriate wrapper to a raw IP block to allow it to work with multiple communication schemes.

Hardware Assist for Real-Time: Reconfigurable logic allows application-specific design of peripheral interface cores. In particular, parts of a software task can be moved to application-specific hardware modules, which might be thought of as intelligent peripheral controllers. Peter Waldeck [8] investigates how real-time performance of a signal processing task can be improved by such a hardware-software codesign approach.

Custom Processors and Peripherals: rSoC allows both processor and peripherals to be customised to support a specific mix of real-time tasks in a way that conventional embedded systems cannot. One of our previous papers [9] explores the range of possible customisations that are possible.

Avionics Applications: The modular approach of Egret can be extended to application specific domains with specific reliability and interface requirements, such as avionics and aerospace applications. We investigate this in [10].

6. Conclusions

This paper very much describes work in progress, and results are limited to an exposition of our research motivations and initial design choices. We are currently working on the development of the Egret platform, which meets this need, and intend to have a working prototype by early 2004.

7. Acknowledgements

The support of the Australian Research Council, and the Xilinx University Programme are gratefully acknowledged.

8. References

[1] Xilinx, "Xilinx FPGA Product Tables," 2003, <http://www.xilinx.com/products/tables/fpga.htm> [accessed 30th Sept., 2003].

[2] Altera, "Excalibur Devices," 2003, <http://www. altera.com/products/devices/arm/arm-index.html> [accessed 30th Sept., 2003].

[3] Atmel, "Field Programmable System Level Integrated Circuits," 2003, <http://www.atmel.com/products/ FPSLIC/> [accessed 30th Sept., 2003].

[4] Triscend, "A7 Configurable System-on-Chip," 2003, <http://www.triscend.com/products/a7.htm> [accessed 30th Sept., 2003].

[5] P. Lysaght, "FPGAs as Meta-Platforms for Embedded Systems," presented at IEEE International Conference on Field Programmable Technology (FPT '02), Hong Kong, 2002.

[6] A.S. Lee, N.W. Bergmann, "Communication Architectures for Reconfigurable System-on-Chip", IEEE Conference on Field Programmable Technologies, Tokyo Japan, December 2003.

[7] T-L. Lee, N.W. Bergmann, "An Interface Methodology for Retargettable FPGA Peripherals", International Conference on Engineering of Reconfigurable Systems and Algorithms, Las Vegas USA, June 2003.

[8] P. Waldeck, N.W. Bergmann, ""Dynamic Hardware-Software Partitioning on Reconfigurable System-on-Chip", International Workshop on System-on-Chip for Real-Time Applications, Calgary Canada, June 2003.

[9] N.W. Bergmann, P. Waldeck, J.A. Williams, "A Catalog of Hardware Acceleration Techniques for Real-Time Reconfigurable System on Chip", International Workshop on System-on-Chip for Real-Time Applications, Calgary Canada, June 2003.

[10] N.W. Bergmann**,** J.A. Williams, "Avionics Upgrade Management using Reconfigurable Logic", Australian International Aerospace Congress, Brisbane, July 2003.