# Egret: a platform for reconfigurable system-on-chip

Neil W. Bergmann[*], John A. Williams[#]
School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane Q 4072, AUSTRALIA

## ABSTRACT

Reconfigurable System-on-Chip (rSoC) design is inherently a complex task with enormous freedom in design parameters such as processor, operating system, and backplane buses. Design efficiency can 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. Egret is an rSoC prototyping platform being developed at the University of Queensland, Australia, and this paper explains and justifies the design decisions for the first version of Egret.

**Keywords:** FPGAs, Reconfigurable Logic, System-on-Chip, Rapid Prototyping

## 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, ambient intelligence or organic computing. As more and more everyday objects become networked, there will be an increasing demand for embedded systems programmers and system designers. Unlike conventional desktop systems, new applications cannot be developed with software alone, but rather application-specific computing hardware and software must be developed for each new application, and this hardware and software must be interfaced to the application-specific sensors, actuators and communication media of the appliance.

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.

Embedded system development becomes significantly more difficult when systems become networked. High-speed network connections introduce considerable hardware and software complexity in the implementation of network standards and protocols. Significant additional complexity is also introduced by the immediate need to add network security, so that embedded networked appliances are protected against unintentional or malicious misuse.

A clear response to the design crisis is to maximise the 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 for use within a single design group. These platforms might include processor choices, standard peripherals, a standard operating system, and even some standard modular circuit boards.

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.

---

[*] n.bergmann@itee.uq.edu.au; Phone +61-7-33651182; Fax: +61-7-33654999; www.itee.uq.edu.au/~bergmann
[#] jwilliams@itee.uq.edu.au; Phone +61-7-33658305; Fax: +61-7-33654999; www.itee.uq.edu.au/~jwilliams

This paper will

- Define rSoC technology,
- Provide a design specification for our platform,
- Explain our processor and operating system choices, and the physical and logical modular structure of the platform, and
- Explain the potential research issues exposed by the platform design.

## 2. RECONFIGURABLE SYSTEM-ON-CHIP

System-on-chip (SoC) technology has evolved as the predominant circuit design methodology for custom ASICs. SoC technology moves design from the circuit level to the system level, concentrating on the selection of appropriate pre-designed IP Blocks, and their interconnection into a complete system. However, modern ASIC design and fabrication are expensive. Design tools may cost many hundreds of thousands of dollars, while tooling and mask costs for large SoC designs now approach $1 million. SoC will be a predominant implementation mechanism for high-volume embedded systems, and SoC design houses are currently developing their own design platforms.

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). For low volume applications, and especially for research and development projects in universities, reconfigurable System-on-Chip (rSoC) technology is more cost effective than ASIC-based SoC.

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.

It is our belief that rSoC can provide an attractive base upon which a cost effective embedded systems design platform can be built. Our Egret platform is based on rSoC concepts.

## 3. PLATFORM SPECIFICATIONS

No platform can be universal in its applicability, even within the domain of reconfigurable computing for embedded systems. Any platform is necessarily a compromise between the requirements of a range of potential applications. Before designing a platform, it is important to decide upon the particular aspects of the platform design which are most important. In other words, it is useful to have a broad specification for the platform before any individual decisions are made.

Because our work is university-based, our requirements are different to those of commercial platforms.

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 (especially undergraduate and coursework masters project 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 some platform specifications. This section explains the specifications, and the subsequent section explains how these specifications are being realised.

## 3.1. Modularity

A modular system is one in which a particular system can be composed from a selection of modules from within a larger general purpose pool of modules. In the case of Egret, we desire modularity in three domains:

- Logical Hardware Modularity – 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 – circuit board modules can be plugged together to meet the particular interfacing, memory, transducer, networking and power supply needs of the system, and
- Software Modularity – software modules can be added to meet device driver, networking, data management and application specific code requirements of the system.

## 3.2. 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.

## 3.3. 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. This configuration will evolve over time:

- Initially, physical modules will be added statically before system boot, and the appropriate hardware and software modules will be added statically to FPGA and operating system configurations.
- Later, this FPGA and OS configuration should happen automatically at system boot time, through automatic module identification and interface negotiation; and
- Finally, the goal would be for automatic “hot swap” capability of hardware modules.

## 3.4. 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.

## 3.5. Simple Design Tool Chain

Our experience of reconfigurable system-on-chip design is that it is complicated, and has a high learning curve. 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. Conversely, experienced designers should be able to make full use of the capabilities of the reconfigurable computing fabric.

### 3.6. Reconfigurability

As far as it 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.

### 3.7. 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. These include:

- Hardware-Software tradeoffs in reconfigurable system-on-chip implementations of real-time systems,
- Flexible on-chip interconnection networks for reconfigurable system-on-chip,
- Flexible, re-usable microprocessor peripheral core designs for reconfigurable system-on-chip, and
- Applicability of reconfigurable system-on-chip for particular application domains, including audio processing, network packet processing, video processing, aerospace, and satellite systems.

### 3.8. Ease of Manufacture

Whilst it is not envisaged that our initial designs will immediately be re-targeted to commercial products, the platform should support an easy and efficient integration of a number of individual modules onto a single circuit board.

This implies that 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)
- Third party softcores, eg. from www.opencores.org.

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.

The requirement for a design tool chain with good System-on-Chip support, plus a desire for medium-level processing power at a low cost, encourages us to adopt a commercial soft-core processor – either Nios or Microblaze. Our existing experience with the Xilinx design tool chain has persuaded us to use the Microblaze processor in our first version of Egret.

### 4.2. Operating System

Embedded systems are often real-time systems, and it was considered highly desirable that our operating system be able to support real-time applications. The major choice here is 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 ported this operating system to Microblaze, and details of the port are available at [11].

Linux (and hence uClinux) is not a real-time operating system, with non-deterministic interrupt latencies and other characteristics that confound hard-real time problems. The integration of a (uC)linux real-time extension system for Microblaze/uClinux is currently underway. We are currently porting the RTAI real-time Linux extensions [12] to Egret.

### 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.

Each connector has 120 pins along one edge. One important aspect is pin density scaleability – we require that the same (or very similar) board and connector architecture support a range of FPGA devices, with varying numbers of user IO.

A system will consist of at least one core module containing the system FPGA with the controlling CPU, and additional peripheral modules.

These resources are distributed in a symmetrical manner across the four edge connectors, thus a module board can be inserted in any orientation, as illustrated in Figure 1. 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).

By appropriate orientations, it is simple to have at least four peripheral cards, each with its own dedicated connections to the FPGA along one side of the card stack.

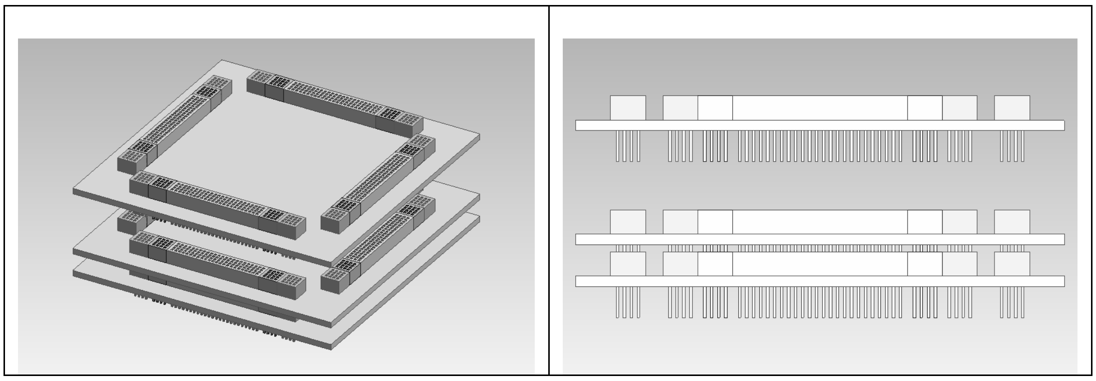

**Figure 1. Visualisation of Egret’s rotationally symmetrical stack-through module architecture.**

Across the platform there 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 allocated available to each module. For example, the architecture may guarantee that modules 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
- 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.

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 standard bus interface to which peripheral chips must be interfaced. 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 off-chip the necessary, and probably lower-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)

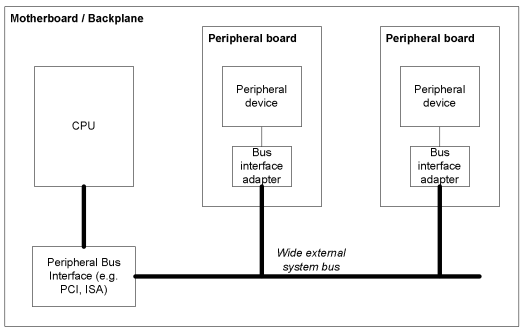

**Figure 2. Typical bus-based system interconnection structure**

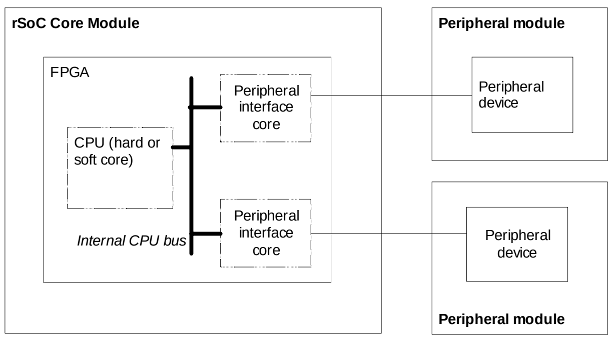

**Figure 3. Egret system interconnection structure with thin peripheral interfaces**

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 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.

One advantage of FPGA-based design is that specially instrumented “debug” versions of device controllers can be used to assist in system development. Also, because all signals are available on every edge of the stack, a debugging header module can be developed which makes every edge signal available for analysis with logic analysers and oscilloscopes etc.

## 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.

### 5.1. 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.

### 5.2. Portable IP Blocks

New interconnection schemes, such as those proposed in section 5.1 above, are most useful if existing IP blocks can be re-used with this new scheme. Tim Lee [7] is investigating an interface adaptor logic scheme, which adds an appropriate wrapper to a raw IP block to allow it to work with multiple communication schemes.

### 5.3. 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.

### 5.4. 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.

### 5.5. 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].

### 5.6. Advanced Reconfigurable Computing

Egret is a natural platform for experimentation into advanced reconfigurable computing topics such as dynamic reconfiguration and self-modifying hardware. Specifically, we are commencing research investigations into the use of uClinux as an operating system for reconfigurable computing.

### 5.7. Future Projects

The use of rSoC as a mainstream embedded system implementation strategy is a largely unexplored research area. We expect many more research questions to be raised as Egret is developed further, and new applications are ported to the platform.

## 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.

Our key results are:

- Embedded and Real-time Systems appear to be a good fit to new rSoC technology, but these application domains are largely unexplored.
- rSoC design is an inherently complex design task, which is complicated by the enormous design flexibility provided by the technology. We are most likely to make progress in this area if we have a well-defined standard rSoC platform, which allows rapid application development without compromising the benefits of rSoC technology.

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.

## ACKNOWLEDGEMENTS

The support of the Australian Research Council, and the Xilinx University Program, is gratefully acknowledged.

## REFERENCES

1. Xilinx, "Xilinx FPGA Product Tables," 2003, <http://www.xilinx.com/products/tables/fpga.htm>
2. Altera, "Excalibur Devices," 2003, <http://www.altera.com/products/devices/arm/arm-index.html>
3. Atmel, "Field Programmable System Level Integrated Circuits," 2003, <http://www.atmel.com/products/FPSLIC/>
4. Triscend, "A7 Configurable System-on-Chip," 2003, <http://www.triscend.com/products/a7.htm>
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, “On-chip interconnection schemes for reconfigurable system-on-chip”, SPIE Conference on Microelectronics: Design, Technology, and Packaging, Perth Australia, December 2003.
7. T-L. Lee, N.W. Bergmann, “Interfacing methodologies for IP re-use in reconfigurable system-on-chip”, SPIE Conference on Microelectronics: Design, Technology, and Packaging, Perth Australia, December 2003.
8. N.W. Bergmann, P. Waldeck, “Hardware support for real-time reconfigurable system-on-chip”, SPIE Conference on Microelectronics: Design, Technology, and Packaging, Perth Australia, December 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
11. J.A. Williams, “Linux on an FPGA?!”, online at http://www.itee.uq.edu.au/~jwilliams/mblaze-uclinux/index.html
12. Milan Polytechnic Department of Aerospace Engineering, “RTAI – Realtime Application Interface”, online at http://www.aero.polimi.it/~rtai/