Swallow: Building an Energy-Transparent Many-Core Embedded Real-Time System

To appear in DATE 2016, Dresden, Germany.
Simon J. Hollis and Steve Kerrison
Department of Computer Science, University of Bristol, United Kingdom
E-mail: harryhollis@cantab.net, steve.kerrison@bristol.ac.uk

***Abstract*—*Swallow* is a many-core platform of interconnected embedded real time processors with time-deterministic execution and a cache-less memory subsystem. Its largest current configuration is 480 × 32-bit processors. It is open-source, designed from the ground up to allow the exploration of flexibility, scalability and energy efficiency in large systems of embedded processors. Further, it enables the behavior of various structures of parallel programs to be explored. It is a proof of concept and design example for other potential systems of this kind. We present the energy transparency features and proportional energy scaling of the system that allows it to be expanded beyond hundreds of cores. We discuss the design choices, construction and novel network implementation of *Swallow*. Currently, the system provides up to 240 GIPS, with each core consuming 71–193 mW, dependent on workload. Its power per instruction is lower than almost all systems of comparable scale. We discuss the challenges associated with efficiently utilizing this system, particularly communication/computation ratios, and give recommendations for future systems and their software.**

I. Introduction

Swallow is a scalable many-core system of multi-threaded real-time processors. Our contributions are the system itself, and the design process used to construct an energy-transparent multi-core research system. An energy-transparent system provides a predictable relationship between software execution and hardware energy consumption.

The *XMOS xCORE XS1-L* micro-architecture [1] is used for the underlying compute and network architecture, providing highly predictable time-deterministic program execution and a low latency network. This is the first example of this architecture assembled at this scale, with a network of 16 cores per *Swallow* board (slice), that can be assembled into systems of much larger sizes, such as in Fig. 1. The current largest demonstrated system is 480 cores. The hardware designs are available under an open source license.

Swallow has been developed as an experimental system for investigating techniques to improve energy efficiency in scalable parallel systems. Key aims of the platform include:

- Scale to hundreds of cores and beyond.
- Deliver proportional scaling in performance and energy.
- Make energy consumption transparent to aid parallel software design exploration for energy efficiency.
- Support a variety of parallel application types and data sharing methods [2], including groups of tasks, pipelines, client/server, message passing and shared memory.

Figure 1. An eight board, 128 core stack of *Swallow* slices.

- Use a real-time embedded system, in contrast to typical heavyweight parallelism of application specific or non-real-time domains.

To support these aims, we built a system that allows flexible expansion and provides energy measurement capabilities. This paper focuses on Swallow and the design challenges surrounding it. A distributed operating system has been developed for *Swallow* [3], and a wider study of benchmarks and program structures for *Swallow* and other platforms is future work.

In this paper, we first examine *Swallow*'s energy measurement capabilities in Section II and its appealing energy proportionality in Section III. Then, we discuss the design considerations and decisions in Section IV, before explaining the novel network implementation in Section V. This is followed by Section VI, a survey of related work, and finally conclusions are made in Section VII.

II. Energy measurement

A number of power measurement points are designed into the system to provide energy-transparency. *Swallow* cores are powered by five separate switch-mode power supplies, each fed from a main 5 V supply. Four of these deliver 1 V to two chips each (four cores), the fifth supplies 3.3 V for I/O, etc.

Each power supply has a shunt resistor on its output and associated probe points. We created a daughter-board that incorporates sensitive differential voltage amplifiers and a high-speed multi-channel analogue-to-digital converter. The resulting system is able to measure individual power supply energy consumption at up to 2 M samples/sec, or 1 M/s if all supplies are sampled simultaneously. Schematics are available online as part of the *Swallow* project open source contribution.

Table I
PER-BIT ENERGIES OF *Swallow* LINKS.

Link type	Data Rate	Max Link Power	Energy per bit
On-chip	250 Mbit/s	1.4 mW	5.6 pJ/bit
On-board, vertical	62.5 Mbit/s	13.3 mW	212.8 pJ/bit
On-board, horizontal	62.5 Mbit/s	12.6 mW	201.6 pJ/bit
Off-board, 30cm FFC	62.5 Mbit/s	680 mW	10880 pJ/bit

A novel feature of this energy measurement is that the measurement data can be collected on the *Swallow* slice itself. In this way, it is possible to create a program that can measure its own power consumption and adapt to the results. Alternatively, the results can be streamed out of the system using an Ethernet interface.

The separate measurement points allow *Swallow* to monitor the balance of energy consumed by the processor cores and external communication links. We present the energy-per-bit for each link in Table I. The system construction and network are described in more detail in Sections IV and V respectively.

The links consume approximately 200 pJ/bit for package-to-package data. The low value can be attributed to the link protocol, which requires only four wire transitions per byte of data. Therefore, the worst case energy usage in communication is half that of a naïve serial or parallel link. Once transmissions go off-board via long flexible cables, the capacitance of those cables becomes the dominating factor for energy, and the energy cost per bit rises by a factor of 50.

Comparing communication costs to computation, profiling of the XS1-L series of processors [4] shows that instructions cause core energy consumption of in the range of 1.0–2.25 μJ at 400 MHz and 1 V, including static power and dependent upon the operations they perform. We can consider this to be 31–70 nJ per bit operated upon. Data movement is therefore relatively inexpensive compared to data processing, before latency is considered.

III. ENERGY PROPORTIONALITY

In order for a many-core system to be scalable, its energy consumption must be both efficient and proportional to the computation it is undertaking. Modern high-performance computing centers consume vast quantities of energy, and energy density is the most important throttle of continued integration of more and more compute power into a fixed space. In this section, we demonstrate how *Swallow* is both energy efficient and energy proportional, and is therefore scalable.

A. Swallow is energy efficient

The processor and memory architecture in *Swallow* are targeted for the embedded application space, where low energy is a primary design goal. A single processor consumes a maximum of 193 mW when active, leading to 3.1 W/slice. Losses in the on-board power supplies and other support logic increase the overall power consumption to approximately 4.5 W/slice (equivalent to 260 mW/core), so a complete 480 core, 30 slice system consumes only 134 W.

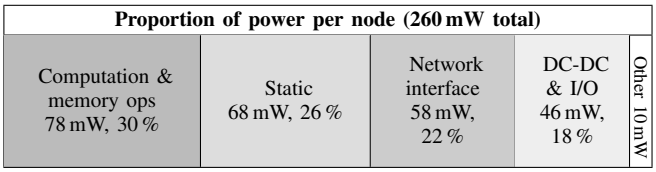

Figure 2. Power distribution for each *Swallow* processor node.

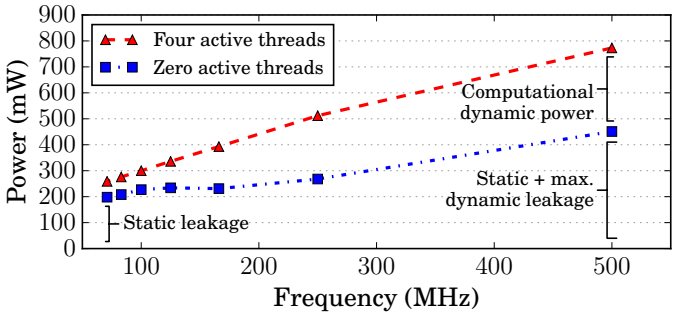

Figure 3. Power consumption with frequency scaling (four cores).

Overall, we see that approximately 18 % of power is used in power supply conversion, support logic and I/O, 30 % is consumed in performing computation, 26 % is spent in non-computational static and dynamic power, and 22 % is used for network interfacing (Fig. 2).

B. Swallow is energy proportional

The XS1-L used in *Swallow* supports dynamic frequency scaling, based on run-time load factors [1]. Fig. 3 shows how the power consumption of a group of four cores scales with their clock frequency.

The power consumed per core, P_c, ranges from 193 mW at 500 MHz to 65 mW at 71 MHz when under heavy load, and 113 mW at 500 MHz to 50 mW at 71 MHz when all cores and threads are idle. The characteristics are linear, giving a directly energy proportional response to clock speed, f, under load,

$$P_c = (46 + 0.30f)\ \text{mW}. \quad (1)$$

Thus, static power is 46 mW and dynamic is 0.3 mW/MHz.

Although the current version of *Swallow* does not support voltage scaling, newer xCORE devices do support full DVFS. The additional power savings from voltage scaling on top of frequency scaling can be reliably calculated from knowing the power formula $P = CV^2f$, where C is the capacitance of the switching transistors and V is V_{dd}. We have determined the minimum allowable voltage experimentally to be 0.6 V at 71 MHz and 0.95 V at 500 MHz, and calculate the equivalent DVFS savings for *Swallow* in Fig. 4.

IV. DESIGN OF SWALLOW

The *Swallow* project sought to build a system from commercially available components, keeping costs and design effort low when compared to building a custom processor or collection of IPs in a bespoke SoC/NoC. This demonstrates that many-core systems can be built and used for research quickly, cheaply and with relatively low risk. The system is

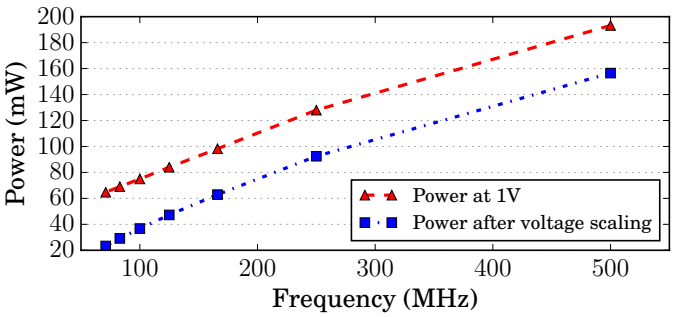

Figure 4. Impact of voltage and frequency scaling on power consumption (four active thread load) for one core.

also designed to be energy-transparent, giving insight into how many-core systems and software consume energy.

A. Processor selection

The choice of processor was governed by a desire to build a system of many predictable embedded processors, upon which novel parallel programming approaches and experiments could be performed, exploiting this predictability. This necessitates the processor have time-deterministic instruction execution in terms of both instruction scheduling and the behavior of the memory hierarchy. The interconnection capabilities must also scale, at least into the hundreds of cores. A number of candidate processors are summarized in Table II.

As is visible in this comparison, few commercial processors present the necessary characteristics of a scalable architecture. The XMOS XS1-L is the only candidate to provide all of these. We further investigated the architecture and determined that its feature set was an excellent match for our system requirements. Key characteristics of XS1 are:

- 64 KiB of single-cycle SRAM local memory per core;
- message passing between cores;
- fast on-chip links, tunable off-chip links;
- fixed instruction completion time for most instructions;
- hardware threads with no context overhead;
- ISA-level primitives for I/O and networking; and
- a network of up to 2^{16} interconnected XS1-L cores.

There are several chips available that are based on XS1-L architecture, with core counts of one or two, with a range of packages and integrated peripherals. To maximize the density of the final configuration a dual-core *XS1-L2A* device was selected. The maximum operating frequency is 500 MHz, yielding 500 MIPS potential throughput per core.

B. Scalable construction

For economic and reliability reasons, we decided to construct *Swallow* from *slices*. A slice is shown in Fig. 5. Each *Swallow* slice comprises sixteen processors across eight chips, with ten off-board network links.

Several components are highlighted in Fig. 5, including the chips, external interconnect headers along all four board edges, along with GPIO near the top edge of the board for further interfaces. The slices are 105 mm × 140 mm in size, with a maximum operating power of 5 W at 12 V. Slices are

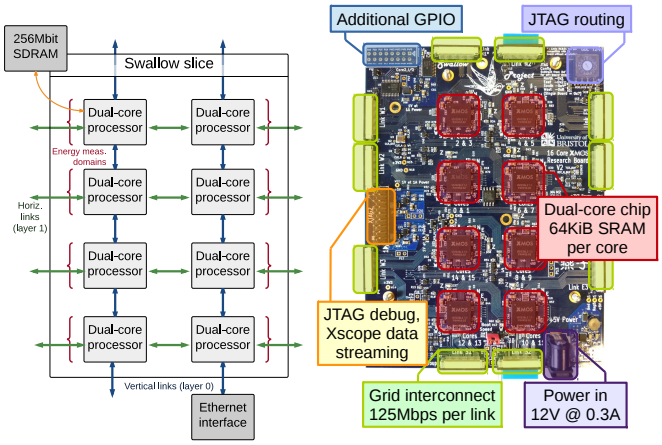

Figure 5. A *Swallow* slice, topology (left); photograph (right), one of forty boards that can be assembled into a grid.

connected with flexible FFC-type ribbon cables. Mounting holes allow vertical stacks to be assembled to minimize the system's footprint (see Fig. 1).

Forty *Swallow* slices have been manufactured, enabling the construction of a 640 processor system. However, yield issues, mostly with edge connectors, mean that the largest machine we have been able to build and test is 480 cores.

C. Computational throughput

Equation (2) shows that the per-thread Instructions Per Second, IPS_t, and processor aggregate throughput, IPS_c, scales according to the number of active threads, N_t, on each core.

$$\text{IPS}_t = \frac{f}{\max(4, N_t)}, \quad \text{IPS}_c = \frac{f \times \min(4, N_t)}{4} \tag{2}$$

This is a property of the XS1-L processor's four-stage execution pipeline [5] with overhead-free thread context switching.

D. Network-on-Chip Implementation

Swallow exploits the network technology of the XS1 architecture. Each XS1-L processor has a number of external communication links. Links are flexibly allocated, with software configurable network partitioning and routing. Links can be aggregated to form a single logical channel with increased bandwidth; or connected separately to different parts of the network, increasing the dimension of the network topology. Links are managed by a switch, with one switch per core in the XS1-L micro-architecture.

V. Swallow Network

Swallow contains a low latency interconnect, suitable for supporting arbitrary traffic types and a mix of parallel or non-parallel applications. In the following section we give an overview of the interconnect and implementation details.

A. Network topology

The device chosen for the *Swallow* system contains two cores and exposes four external network links, in the way shown in Fig. 6. The internal links have four times more

Table II
COMPARISON OF CANDIDATE SWALLOW PROCESSORS. ONLY THE XS1-L MEETS ALL REQUIREMENTS.

Processor	Features				Requirements	
	Cores × data width	Super-scalar	Cache	Typical memory configuration	Multi-core interconnect	Time deterministic
ARM Cortex M	1×32-bit	No	Optional	<varies>	No	W/o cache
ARM Cortex A, single core	1×32-bit	Yes	Yes	<varies>	No	No
ARM Cortex A, multi-core	4×32-bit	Yes	Yes	<varies>	Coherent mem.	No
Adapteva Epiphany	64×32-bit	Yes	No	Local + global SRAM	NoC + external	No
XMOS XS1-L	1×32-bit	No	No	Unified, single cycle SRAM	NoC + external	Yes
MSP430	1×16-bit	No	No	I-Flash + D-SRAM	No	Yes
AVR	1×8-bit	No	No	I-Flash + D-SRAM	No	No
Quark	1×32-bit	No	Yes	Unified DRAM	Ethernet	No

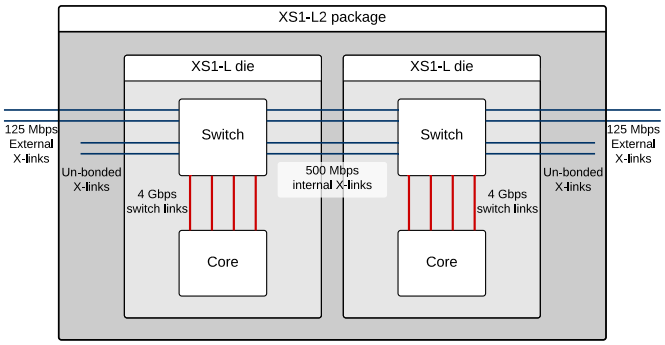

Figure 6. Network links of the XMOS XS1-L2 in a single *Swallow* node.

bandwidth than external links. Data words can be transferred from the core to the network hardware with *just three cycles* of latency (6 ns). This compares with 80 ns for the BlueGene/Q system [6]. In *Swallow*, four external links are then arranged to connect `North`, `South`, `East` and `West` to other devices.

An interesting artifact of *Swallow*'s device selection is that, as is evident in Fig. 6, it is not possible to make a conventional 2D mesh topology. The internal links are already utilized by core-to-core connections. This, combined with the pin-out of the package, means that the most effective grid-like structure is an *unwoven lattice*, shown in Fig. 7. This presents interesting routing challenges, requiring a unique strategy.

The unwoven lattice network is effectively composed of two layers, with each layer containing half of the available cores. One layer routes in the vertical dimension and the other layer routes in the horizontal dimension. Each node in the network also has a connection to a node in the opposite layer, which takes place within a chip package. This topology requires that two-dimensional routes be translated into a form of 2.5 D routing, where routing between layers is required to change horizontal/vertical direction. The dimension order routing [7] strategy that we use prioritizes the vertical dimension first. If a node is attached to the horizontal layer and a vertical communication is required, the message must therefore be sent to the other layer first. In this scheme, there will be at most two layer transitions; the exemplary case being two nodes attached to the horizontal layer that do not share the same vertical index.

Links between *Swallow* slices use flexible cables, allowing the physical topology of the network to be adjusted across a wide variety of configurations, further extending the range of experiments that can be carried out. New routing algorithms can simply be programmed in software to cope with these.

B. Network implementation

Each processor node in the XS1-L2 contains one core and one switch, which has four internal links and two external links, as per Fig. 7. Switches use wormhole routing with credit-based flow control. The instruction set abstracts the network into channel communication which can take the form of either channel switched or packet operation.

Routes are opened with a three byte header prefixed to the front of the first token emitted from a channel end. Any network links utilized along the route are held open until the source channel emits a closing control token. If the close token is never emitted, links are permanently held open, effectively creating a dedicated circuit between two endpoints. The overhead of packet data reduces throughput to approximately 87 % of the link speed, but is dependent upon the packet size.

Multiple links can be assigned to the same routing direction, where a new communication will use the next unused link. This increases bandwidth, provided the number of concurrent communications is equal to or greater than the number of links. This is particularly exploitable for on-chip communication, where there are four links between each core. Provided no more than three links are used for channel switching, packeted data can still flow through the network.

C. Network details

On- and off-chip links all use the same five wire protocol [1], but run at different speeds to preserve signal integrity. The connections and speeds used in *Swallow* are shown in Fig. 6. There is a maximum throughput of 500 Mbit/s per internal link and 125 Mbit/s per external link, providing 2 Gbit/s in-package and 500 Mbit/s externally.

Links send data in eight-bit tokens comprised of two-bit symbols. A token's transmit time is $3T_s + T_t$, where T_s is the inter-symbol delay and T_t the inter-token delay, measured in clock cycles. The fastest possible mode is $T_s = 2, T_t = 1$, yielding the aforementioned 500 Mbit/s at 500 MHz.

The total core-to-core latency for an eight-bit token is 270 ns. The total core-to-core latency for a 32-bit word between packages is 360 ns, equivalent to the time taken to for

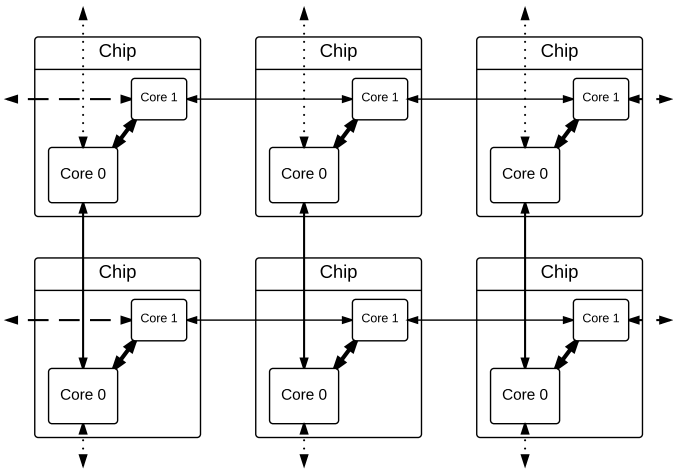

Figure 7. *Swallow*'s unwoven lattice network topology.

the sending thread to execute 45 instructions. Between two cores in a package, this reduces to 40 instructions. Core-local communication takes 50 ns, or approximately 6 instructions.

D. Ratio of communication to computation

The ratio of computation to communication is important to the performance of all systems, where a processor's ability to process data is governed by how quickly data can be moved through the system. This ratio has been termed $\frac{C}{C}$ in some work [8], but for clarity, we adopt the notation $\frac{E}{C}$ [9], where E is *execution* or computation, and C is communication. Communication can be considered movement of data to/from memory, or messages over a network. We define the scope of the ratios (single cores or cross-system) as appropriate.

On *Swallow*, it is possible for a single thread of execution to issue 125 MIPS. Instructions operating on 32-bit data give a maximum per-thread communication throughput of 4 Gbit/s. With four or more threads active threads, $E = 16$ Gbit/s. Core-local communication can sustain this data rate, such that $E = C$ and therefore $\frac{E}{C} = 1$. However, where communication is not core-local, worst case C becomes 250 Mbps and 62.5 Mbps for internal and external links respectively. Communication instructions will block if the output buffer is full.

The total bandwidth of all four package-internal links, $C = 1$ Gbps, gives $\frac{E}{C} = 16$. External links are a quarter of the bandwidth, so externally this ratio increases to 64, assuming non-contended links. If four threads contend for one link, $\frac{E}{C}$ becomes 256.

Considering a slice of sixteen cores, if we take the vertical bisection bandwidth, then $C = 250$ Mbps. If all available compute resource attempts to communicate over the bisection, then $E = 128$ Gbps and therefore $\frac{E}{C} = 512$, which is undesirable. The impact of this on a full *Swallow* system depends on the arrangement of slices and, most importantly, the communication patterns of the programs running upon it.

The systems described in Section VI have system wide computation to communication ratios ranging from 0.42 to 55. Larger, memory-oriented many-core processors such as the Xeon Phi achieve analogous ratios for FLOPS/memory word that sit in the top half of this range [10]. Based on the highly desirable core- and chip-local ratio, and the potential impact of contention over external links, the following considerations should be made when running applications on *Swallow*:

- Prefer core-local communication where possible; it is low latency, high bandwidth, with tight timing predictability.
- Chip-local communication should be the next preference. Link reservation through channel switching can allow similar levels of predictability, with a potential impact on any external traffic routed through the chip.
- Off-chip communication is the most contentious, with the least predictability in large systems, particularly with complex communication patterns.

These problems are analogous to issues in memory hierarchy of more conventional multi-core systems. However, in *Swallow*, more control is placed in the hands of the developer, where the allocation of work onto threads and cores, combined with sensibly scheduled, mostly localized communication, allowing predictability and good $\frac{E}{C}$ to be achieved.

E. External network interface

Communication into and out of a *Swallow* system is performed over an Ethernet bridge module. This module attaches to the *Swallow* network and is addressable as a node in the network, but forwards all data to and from an Ethernet interface. Using this bridge, it is possible to both load programs[1] into and stream data in/out of *Swallow* over Ethernet. *Swallow* supports up to two Ethernet modules per slice (on the `South` external links). Each bridge can support up to 80 Mbit/s of full-duplex data transfer.

VI. Related Work

There are few embedded systems made at the same scale as *Swallow*, with even fewer designed for general purpose computation. Here we identify noteworthy examples, and in Table III provide a comparison of scaling, technology and power characteristics of a number of recent systems.

The Tilera Tile [13] comes the closest to matching *Swallow*'s goals and form. The Tile64 is a 64-core device with five overlaid networks to provide low latency and high throughput between cores in a software configurable way. The effect is to provide a very agreeable computation to communication ratio of 2.4 with 64 cores, and general purpose computation is supported as well as sophisticated network traffic manipulation. The system is highly optimised for streaming traffic, but relies on adding additional networks to improve network performance in larger systems. Tilera's 64 core device [15] consumes 300 mW/core (19.2 W/device).

Adapteva's specialized floating point Epiphany [14] architecture has a similar grid structure to Tilera Tile, with three independent networks. It requires less than 2 W for a 28 nm 64-core device (31 mW/core).

The Centip3De system [12] aims to use 3-D stacked dies to implement a 64-core system based on the ARM Cortex-M3

[1] Swallow boot video: https://youtu.be/kUo11tTeYK0

Table III
COMPARISON OF SCALE, TECHNOLOGY AND POWER PROPERTIES OF RECENT MANY-CORE SYSTEMS.

System	ISA	Cores / chip	Total cores	Tech. node	Power / core	Frequency	µW/MHz
Swallow	XS1	2	16–480	65 nm	193 mW	500 MHz	300
SpiNNaker [11]	ARM9	17	1,036,800	130 nm	87 mW	200 MHz	435
Centip3De [12]	Cortex-M3	64	64	130 nm	203–1851 mW	20–80 MHz	2540–2300
Tile64 [13]	Tile	64	64–480	130 nm	300 mW	1000 MHz	300
Epiphany-IV [14]	Epiphany	64	64	28 nm	31 mW	800 MHz	38.8

processor. Whilst its scale is within an order of magnitude of *Swallow*, it relies on a series of crossbars and coherent DRAM storage and thus scalability to larger sizes may be restricted. Further, the design choices leave it with an undesirable computation to communication ratio of 55. Centip3De exploits near-threshold computing in 130 nm, small ARM Cortex-M3 cores and consumes 203–1851 mW/core, depending on its configuration. Centip3De's high power is mainly due its cache-centric design, which is not present in *Swallow*.

The SpiNNaker system [11] is the best provisioned system in the large scale. SpiNNaker, like Centip3De is based on ARM cores, connecting up to one million ARM9 parts via a highly-connected network. However, the system is targeted at solving a single problem, making it very difficult to overlay general computation tasks, and also making it hard to draw parallels with *Swallow* and the other systems mentioned above. It dissipates an average of 87 mW per core. It is more densely integrated than *Swallow*, with 17 cores per chip. If *Swallow* had these economies of scale, an equivalent level of power efficiency is very possible to obtain, however this would require the fabrication and packaging of new processors.

Swallow's power per core is in the middle of the surveyed range, in line with its operating frequency and process node with respect to the other systems.

VII. CONCLUSIONS

We have presented a many-core real-time embedded system, named *Swallow*, that has been demonstrated on a scale of up to 480 cores, and can be scaled into the thousands. The purpose of *Swallow* is to explore the challenges of building a networked many-core system with predictable embedded processors, and to demonstrate that such a feat is possible. It is being used to investigate energy efficiency of parallel programs and the impact of computation to communication ratios of such systems on these programs. *Swallow* demonstrates excellent core- and chip-local network latency and bandwidth, allow predictable program behavior. The effects of slower, more contended links can be mitigated through appropriate user-controlled communication patterns and allocation of resources.

Swallow is energy efficient, using only 193 mW/core with four active threads. XS1-L processors support dynamic frequency scaling and this allows an energy reduction to as little as 50 mW/core when idle. This could be lowered further in a future revision, by using configurable power supplies.

The platform serves as a new case study and data source in the spectrum of many-core systems, being the first with a strong focus on using real-time, deterministic general purpose embedded hardware.

OPEN SOURCE RELEASE

Swallow is an open source project released in several parts, with licenses appropriate for the hardware, operating system, supporting tools and application software. The latest status of all releases can be found at https://swallow-project.github.io.

ACKNOWLEDGMENT

The authors would like to thank Jamie Hanlon for his advice and technical discussions. The initial 10 prototypes of the *Swallow* system were kindly sponsored by XMOS Ltd. The full system was funded by the University of Bristol's research *Pump-priming* scheme. Research conducted on the *Swallow* platform and the presentation of this work has received funding from the European Union 7th Framework Program agreement no 318337, ENTRA - Whole-Systems Energy Transparency.

REFERENCES

[1] D. May *et al.*, "XS1-L System Specification," pp. 1–40, 2008.
[2] J. Diaz *et al.*, "A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, pp. 1369–1386, Aug. 2012.
[3] S. J. Hollis *et al.*, "nOS: a nano-sized distributed operating system for resource optimisation on many-core systems," Tech. Rep., 2015.
[4] S. Kerrison *et al.*, "Energy Modeling of Software for a Hardware Multithreaded Embedded Microprocessor," *ACM Transactions on Embedded Computing Systems*, vol. 14, pp. 56:1–56:25, Apr. 2015.
[5] XMOS, "XS1-L16A-128-QF124 Datasheet," 2014.
[6] S. Kumar, "Challenges of Exascale Messaging Library Design: Case Study with Blue Gene Machines," in *CASS Workshop*, 2012. [Online]. Available: http://www.ccs3.lanl.gov/cass2012/talks/kumar.pdf
[7] H. Sullivan *et al.*, "A large scale, homogeneous, fully distributed parallel machine, I," in *Proceedings of the 4th annual symposium on Computer architecture - ISCA '77*. New York, New York, USA: ACM Press, May 1977, pp. 105–117.
[8] M. Crovella *et al.*, "Using communication-to-computation ratio in parallel program design and performance prediction," in *Proceedings of the Fourth IEEE Symposium on Parallel and Distributed Processing*. IEEE, 1992, pp. 238–245.
[9] D. May, "Communicating Processors: Past, Present and Future," in *Networks-on-Chip, International Symposium on*, 2008. [Online]. Available: http://www.cs.bris.ac.uk/~dave/nocs.pdf
[10] K. S. Solnushkin, "Memory Bandwidth for Intel Xeon Phi." [Online]. Available: http://clusterdesign.org/2013/02/memory-bandwidth-for-intel-xeon-phi-and-friends/
[11] S. Furber *et al.*, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, pp. 2454–2467, 2013.
[12] D. Fick *et al.*, "Centip3De: A cluster-based NTC architecture with 64 ARM cortex-M3 cores in 3D stacked 130 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 104–117, 2013.
[13] S. Bell *et al.*, "TILE64^{TM} processor: A 64-core SoC with mesh interconnect," in *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, vol. 51, 2008.
[14] Adapteva, "E64G401 EPIPHANY^{TM} 64-core microprocessor datasheet." [Online]. Available: http://www.adapteva.com/docs/e64g401_datasheet.pdf
[15] A. Agarwal *et al.*, "The Tile Processor Architecture, Embedded Multicore for Networking and Multimedia," in *Hot Chips*, 2007.