TU/e technische universiteit eindhoven

Ladon — A 24-processor Low-Power Performance-Scalable Processor Module for Sensor Platforms

Phillip Stanley-Marbell

ES Reports
ISSN 1574-9517

ESR-2008-05
24 January 2008

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems

People who are really serious about software should make their own hardware. — Alan Kay

http://www.es.ele.tue.nl/esreports
esreports@es.ele.tue.nl

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems
PO Box 513
NL-5600 MB Eindhoven
The Netherlands

Ladon — A 24-processor Low-Power Performance-Scalable Processor Module for Sensor Platforms

Phillip Stanley-Marbell
Technische Universiteit Eindhoven
Den Dolech 2, 5612 WB Eindhoven, The Netherlands

Abstract

Node architectures in wireless sensor networks often have to balance ultra-low-power idling, with an occasional need for high performance, due in part to the sporadic nature of the phenomena they usually monitor. Higher performance processors, with more computing resources, peripherals, and larger on-chip memories, typically also have larger idle power dissipation. Existing node platforms employing off-the-shelf microcontrollers target one point in a tradeoff space, and are either resource constrained and low power, or are high performance and less power efficient.

Rather than employing a single processor implementation that is limited to one point in the tradeoff space, an alternative approach is the use of multi-microcontroller designs. This is feasible since many sensor network applications have a fair amount of coarse-grained parallelism, ranging from protocol stack pipeline parallelism to event-level concurrency. Motivated by these observations, and by recent attempts in the research literature to develop programming models, language implementations, and system software for platforms consisting of networks of resource-constrained computing elements, a hardware platform, Ladon, was developed. The platform implements a miniature reusable multi-microcontroller module for use in wireless sensor network research. This paper presents the system architecture of the Ladon processor module and evaluates its hardware implementation.

1 Introduction

Due to the need to operate for long periods without replenishment of their energy resources, wireless sensor network node platforms employ energy-efficient designs. The power dissipation of a node platform consists of many parts, from the radio interface which usually dominates, to the system's processing element and sensors. As they are often used to monitor infrequently occurring phenomena, nodes spend a majority of their lifetime idle, typically in one or more low-power-dissipation *sleep modes*. Upon the occurrence of an event, e.g., detection of sudden temperature elevation in a fire monitoring application, they must immediately transition to a state where they may perform a significant amount of processing and communication. The processing tasks in such situations may range from simple control decisions, to the execution of complex algorithms for processing the signals from sensors.

The two modes of operation — deep sleep and heavy computation — are in some ways at odds with each other. On one hand, reducing the power dissipated during sleep requires a minimization of the processor integrated circuit size[1], to reduce leakage power dissipation. On the other hand, providing high performance computation requires more sophisticated architectures implemented in a greater number of transistors, larger on-chip memories, and so on. Such designs however have a larger idle power dissipation. This is due to a combination of the use of smaller process geometries and their associated lower process threshold voltages and higher leakage, as well as to the use of larger numbers of transistors. To illustrate, Table 1 shows the operating voltage range, minimum *sleep mode* power consumption, and theoretical maximum instruction throughput for a representative selection of state-of-the-art embedded processors. In the table, the sleep mode shown for a given architecture is the lowest-power mode at which it retains executing program state. Figure 1 shows the data from Table 1 extrapolated to systems employing multiples of each processor implementation, assuming applications can be made to

[1] For a given implementation process technology and circuit-level techniques such as the use of multiple threshold voltages, *body-biasing*, and *high-K dielectrics*.

Table 1. Trends in peak performance versus idle power for a spectrum of contemporary 8–32-bit microcontrollers. Performance characteristics of the theoretical maximum instruction throughput is shown normalized to the performance of an 8-bit architecture, with the simplifying assumption of 16- and 32-bit architectures having twice and quadruple the instruction throughput of an 8-bit, architecture respectively.

Microcontroller	Operating Voltage Range	Min. State-Retention Sleep Current	Normalized Max. Performance
Freescale MC9S08GT60 (HCS08)	1.8 V–3.6 V	0.02 μA @ 2 V, 25 °C	20 M instrs. / s
ATMEL ATMEGA128L	2.7 V – 5.5 V	0.2 μA @ 3.3 V, 25 °C	16 M instrs. / s
Microchip PIC18F2X1X	2.0 V – 5.5 V	0.1 μA @ 3.3 V, 25 °C	10 M instrs. / s
TI MSP430F2274	1.8 V–3.6 V	0.1 μA @ 2.2 V, 25 °C	16×2 M instrs. / s
TI MSP430F1611	1.8 V–3.6 V	0.2 μA @ 2.2 V, 25 °C	8×2 M instrs. / s
Microchip PIC24FJ128GA010	2.0 V – 3.6 V	3.0 μA @ 2.0 V, 25 °C	16×2 M instrs. / s
ATMEL AT91SAM7S256 (ARM7)	1.85 V core, 3.3 V I/O	38.3 μA, 25 °C	55×4 M instrs. / s
Freescale i.MX31 (ARM11)	1.65 V core, 3.3 V I/O	250 μA, 25 °C	532×4 M instrs. / s

take advantage of the multiple processing elements.

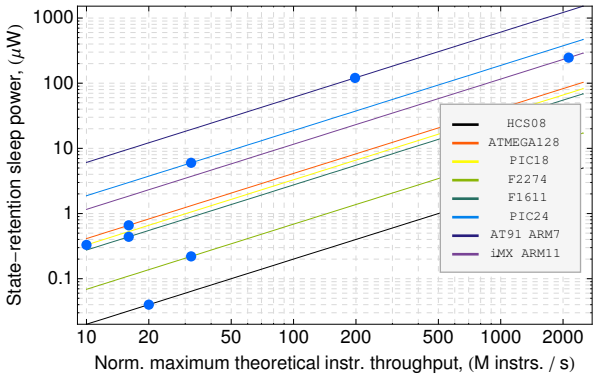


Figure 1. Visualization of trends in peak performance versus idle-power dissipation for a selection of contemporary microcontrollers. The overlaid lines correspond to the extrapolated idle power-performance points, that would be attained when employing multiple processing elements of a given type.

From Figure 1 and Table 1, it can be seen that low-performance microcontrollers, such as the HCS08 family of devices, typically have much lower idle power dissipation than high performance microcontrollers. The difference in this sleep mode power dissipation, between the lowest power dissipation (HCS08) and highest performance (AT91SAM7S and i.MX31) microcontroller in Table 1, is between three and four orders of magnitude[2].

These observations motivate the investigation of a middle ground between the extremes in the idle-power versus high-performance tradeoff space — a *performance-scalable* architecture, capable of delivering both *low idle power* and *high performance*. Ideally, such an architecture would provide computation resources (e.g., multiple processor cores) as needed, shutting down resources when they are not needed, within a single integrated circuit. With appropriate care, the benefits of such an architecture can be gained even with a printed-circuit-board (PCB) *multi-microcontroller* implementation.

In practice, the benefits of such systems will depend on the ability to construct efficient interconnect topologies for such designs, with low implementation cost, low hardware overhead, and high performance. In addition to the advantages of different architectures apparent from Table 1, the choice of which microcontroller to use as the basis for such a system will be influenced by system considerations such as the availability of high speed communication interfaces. Lastly, the performance of the resulting system will depend on the ability of software to take advantage of the hardware resources.

To explore these ideas of multi-microcontroller, performance-scalable sensor node platforms, we designed and implemented a reusable processor module, Ladon[3], for use in wireless sensor network applications. The research was motivated by the foregoing observations, and by recent attempts in the literature to develop programming languages and system soft-

[2]Although in general, the higher performance architectures have higher idle power consumption, the Freescale i.MX31 processor implementation provides a better performance-idle power tradeoff than the ATMEL AT91SAM7S256 and Microchip PIC24FJ128GA010 implementations.

[3]Ladon is a multi-headed dragon from Greek mythology.

ware [8] for networks of resource-constrained programmable elements. The Ladon processor module comprises 24 low-power microcontrollers interconnected in an efficient topology. The interconnect forms a directed graph to facilitate the low-power idling of nodes until they have incoming communications. The network topology is a degree-2 Kautz network topology [12], which has many properties making it suitable for hardware implementations such as the work described in this paper. By intelligently mapping the logical interconnect topology to the built-in hardware resources of the component microcontrollers, the implementation is realized with minimal software overhead, and without the need for additional logic outside the 24 processors. Following an overview of related research in Section 2, an overview of the system architecture is presented in Section 3. Section 4 details the hardware realization of the architecture, and Section 5 presents an evaluation of its performance. The paper concludes in Section 6 with a summary and directions for future research.

2 Related Research

A large variety of wireless sensor node platforms have been developed over the last decade. These platforms encompass the range from miniature nodes, usually employed for application-specific purposes [11, 28, 19], to general purpose resource-constrained platforms [11, 20], to high performance node platforms [3, 16, 1, 14, 7].

To enable the construction of customized node platforms from a standardized set of components, a number of modular sensor node architectures have been proposed [24, 21, 13, 2, 4, 15]. While some of these modular architectures employ multiple processing elements, either to enable power-performance tradeoffs [24, 15], or to facilitate communication over a high-speed interconnect bus [13], the number of processing elements they incorporate is always small, with the most processing elements being four [13]. In contrast to these platforms, the work described in this paper focuses on a reusable multi-processor module, that may be used as the basis either for a monolithic performance-adaptive sensor platform, or as part of a modular system architecture. Unlike in modular platforms, in which the goal is to be able to employ an interchangeable set of general-purpose modules (sensor modules, radio modules, processing modules, and so on), the focus of the work presented in this paper is on a performance-scalable processing engine. Unlike in modular platforms where shared multi-drop buses

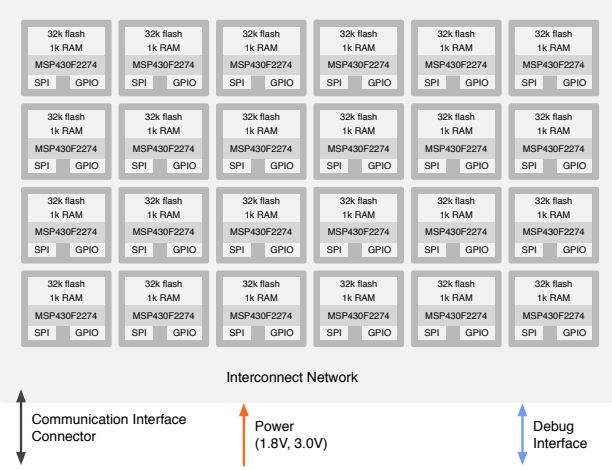


Figure 2. Functional organization of Ladon module system architecture.

are used for communication, to enable easy interchange of peripheral modules, the design of a self-contained *processing module* does not require such flexibility, and we employ a dedicated point-to-point communication interconnect.

The use of computational platforms with multiple processors in resource constrained systems has recently begun to gain interest. The current state-of-the art in such processing platforms is the *Propeller* microcontroller from Parallax [18], which contains eight processing elements communicating over a time-division-multiplex bus. To address the computation challenges peculiar to wireless sensor networks, several custom processor architectures [5, 10] as well as custom processor implementations [27] have been developed. The work presented in this paper provides a concrete platform that demonstrates the possibilities of multi-core architectures for low-power resource constrained systems such as sensor networks. While not a single-chip implementation like the Propeller, it provides more flexibility than Propeller as a research tool, as the interconnect topology it employs has many graph-theoretic properties that enable its use in emulating many others.

3 System Architecture

The Ladon processor module consists of a network of 24 low-power microcontrollers, interconnected in a fixed network topology[4]; the functional

[4] In what follows, the terms *microcontroller, processor, processing element, node* and *vertex* will be used interchangeably if the

organization of the system is illustrated in Figure 2. The processors communicate over multiple point-to-point links rather than a shared bus, increasing inter-processor communication bandwidth, and minimizing the number of processors that may be woken up by, or may need to monitor, ongoing communications. The multiple point-to-point links are in the form of a *Kautz network topology* [12], which has many properties making it a good match for hardware implementations of low-latency, fault-tolerant interconnect networks.

The following section provides an overview of Kautz network topologies and their properties, in the context of the implementation presented in this paper. Section 3.2 follows, with an overview of the mapping of edges in the topology to hardware interfaces on each of the 24 processors in the Ladon module. The ability of all processing elements to remain in their lowest-power modes until they must engage in computation or communication, made possible by properties of the topology and its mapping to hardware in our implementation, is described in Section 3.3. It is followed in Section 3.4 by a description of the efficient and deterministic inter-processor communication made possible by the Kautz topology.

3.1 Kautz Networks

A Kautz network topology is a *directed graph (digraph)*, $G = (V, E)$ with vertex set V and edges E. The vertices correspond to computation nodes (in our case, to processors in the Ladon module), and the edges correspond to communication links (in our case, interconnection links in the module hardware). A Kautz digraph, $K(d, k)$ is *degree-regular* with degree d (all nodes have the same *degree* or number of neighbors, d) and a *diameter* (maximum distance between any pair of vertices) of k. A Kautz network $K(d, k)$ has $N = d^k + d^{k-1}, \forall d, k \in \mathbb{N}$ nodes.

Each node in the graph can be given a label, a string of length k, taken from the alphabet $S = \{0, \ldots, d\}$, with no two adjacent letters in the label being identical. Thus, for example, for a $K(2, 2)$ Kautz network (Figure 3), valid node labels are {01, 02, 10, 12, 20, 21}, however 00, 11 and 22 are not valid node labels (two adjacent letters in the string are identical).

The neighbors at the end of outgoing edges of a node in the graph topology can be determined by shifting its leftmost letter out, and shifting in a new rightmost letter, obeying the letter adjacency rules. Thus, for example, the node with label 10 in Figure 3 has outgoing edges to 01 and 02. Similarly, the node 10 has incoming edges from 21 and 01. These properties for constructing the network topology admit simple methods for routing data in a Kautz topology, as will be detailed in Section 3.4.

intention is clear from the context.

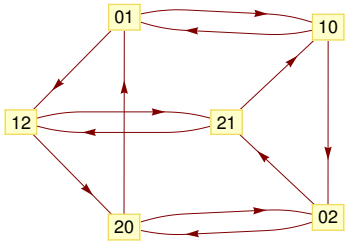


Figure 3. A $K(2, 2)$ Kautz network. It has a diameter (maximum distance between any pair of nodes) of two, and contains at least two disjoint (non-overlapping) paths between any pair of nodes.

Table 2 compares properties of Kautz networks to those of other commonly used interconnect topologies, such as fully-connected networks (all nodes directly connected), trees, meshes and linear arrays (a form of which is used in the Propeller microcontroller [18]). For a given number of nodes (and hence a given peak-performance versus idle power tradeoff point for a multi-microcontroller system, Figure 1), the *diameter* is an indicator of the worst-case inter-node communication latency. Similarly, the *edge bisection bandwidth*, the number of edges (links) that must be deleted to divide the topology into two non-communicating halves, is an indicator of the capacity of the network to support random communication patterns.

Kautz networks have many interesting properties which make them well suited for interconnection networks, particularly when employing a hardware implementation of the network topology:

- All nodes in the Kautz network have the same degree. This makes mapping the network topology to a hardware implementation uniform.
- In contrast to, say, a fully connected topology, Kautz topologies admit easy mapping to a two-dimensional layout with non-intersecting links.
- A Kautz network $K(d, k)$ has a diameter of k. Low diameter means a low *worst-case* number of hops when nodes that are not direct neighbors must communicate.

Table 2. Comparison of the properties of Kautz networks against a range of other commonly-used network topologies.

Topology	Number of Nodes	Diameter	Edge Bisection Bandwidth
Linear array	N	$N-1$	1
Tree	N	$\log(N)$	1
Mesh	N	$2 \cdot \left(\sqrt{N}-1\right)$	$\sqrt{N}$
Kautz $K(d,k)$	$N = d^k + d^{k-1}, \forall d,k \in \mathbb{N}$	k	(See[22] for bounds)
Fully connected	N	1	$N^2/4$

- For any Kautz network $K(d,k)$, there are d non-overlapping paths between any two nodes. This provides fault tolerance, and may also be employed to improve performance, e.g., by splitting outgoing communications over non-overlapping paths.
- Next-hops in routes can be computed easily, in constant time, from current and destination node labels.
- Efficient one-to-all and all-to-all broadcast, as well as all-to-all scatter algorithms exist for Kautz networks [23]. This facilitates the efficient implementation of many programming models over a network of processing elements connected thereby.
- Kautz networks have a high *edge bisection bandwidth*.

The Ladon module employs a $K(2,4)$ topology (Figure 4), to best match the number of communication interfaces (two hardware and two software serial peripheral interface (SPI) ports) available on the component microcontrollers employed in the implementation. The topology has a maximum of 4 intermediaries between any two processing elements.

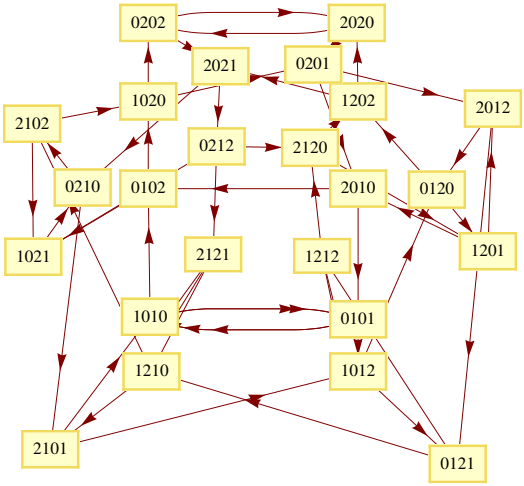


Figure 4. Logical interconnection topology of the 24 processing elements in the Ladon module.

3.2 Link architecture, and mapping Kautz topology to hardware

The nodes in the logical Kautz topology shown in Figure 4 correspond to processing elements in the Ladon module, and links were mapped to communication interfaces on the processing elements.

Each processing element is a TI MSP430F2274 microcontroller, with 32 kB of flash memory and 1 kB of RAM. The flash memory may be used for both code and data storage, and programs can be executed while resident in flash (with program variables residing in RAM). This particular member of the MSP430 family was chosen because it provided the best trade-off of available computing and memory resources to package size (6 mm×6 mm). Small package size and number of pins were important to be able to employ a large number of processing elements, while at the same time keeping the size of the overall module small. The use of a member of the product family with a smaller pin count also had the added benefit of having slightly smaller idle/leakage power consumption than larger members of the MSP430 family.

To determine a placement of the processing elements that minimized the length of the interconnect lines between any two processing elements in the hardware implementation, the logical node interconnections were provided as input to the `HighDimensionalEmbedding` algorithm in Mathematica. This algorithm embeds the logical graph topology in a higher-dimensional space, followed by projection to a two-dimensional space [9]. While it

in practice provides worse results than other graph-layout algorithms for many common graphs, we found it to provide the best placement of nodes for mapping to a circuit layout with non-intersecting edges.

The edges in the Kautz network were mapped in hardware to serial peripheral interface (SPI) communication ports. Compared to other communication interfaces such as inter integrated circuit communication (I2C), which provides typical communication data rates of 400 kb/s, SPI can provide bit rates of up to 10 Mb/s. The performance of all such interface technologies is however limited by the electrical properties of the communication links, in particular, the capacitance and hence the length of the interconnect wires. Each edge in the logical topology graph was mapped to a separate SPI port (i.e., point-to-point links) to reduce the load capacitance on each wire. Outgoing edges in the graph were mapped to SPI *master* configurations, while incoming edges were mapped to SPI *slave* configurations. This removed the need for additional hardware for SPI slaves to alert masters of ready data, or for master nodes to poll slaves for available data (since SPI slaves cannot initiate communication). The approach also enabled the maximization of the time that idle processing elements spend in a deep sleep mode, reducing power consumption, as described in Section 3.3. Due to the limited number of SPI ports on each device (two hardware SPI ports are available on the MSP430F2274), and the number of outgoing edges at each node (two outgoing edges, as can be seen in Figure 4), a software implementation of SPI over general purpose I/O (GPIO) pins was employed for the second outgoing edge / incoming edge pair. In Section 5, we show that compared to the hardware-driven SPI interfaces, which achieve data rates of up to 8 Mb/s in our implementation, these software-driven links achieve lower but still respectable data rates of up to 1 Mb/s. The mapping of logical topology edges to hardware interfaces is shown in Figure 5.

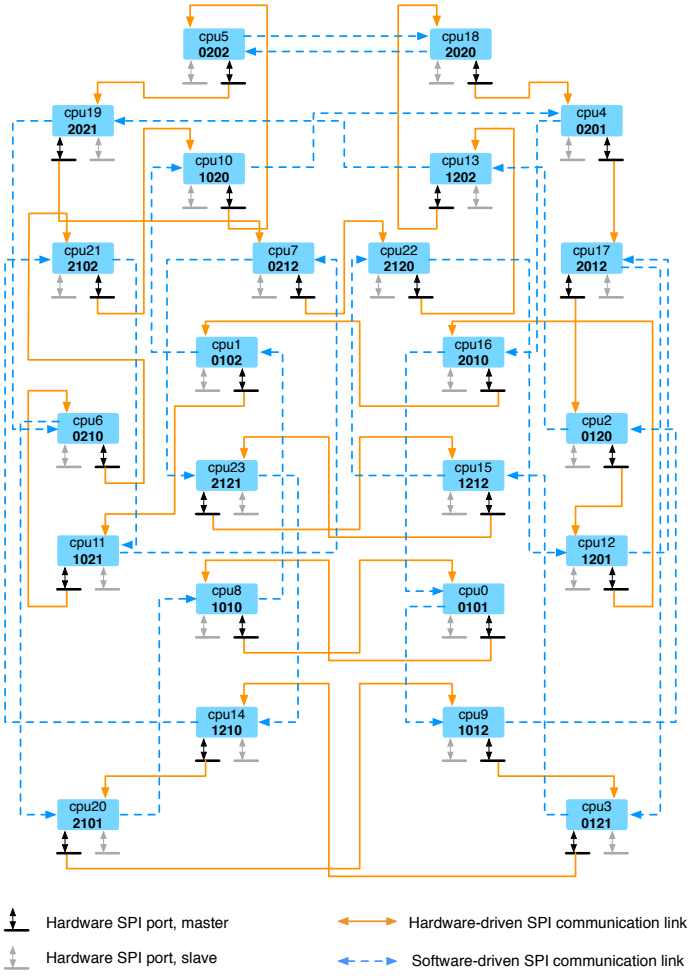


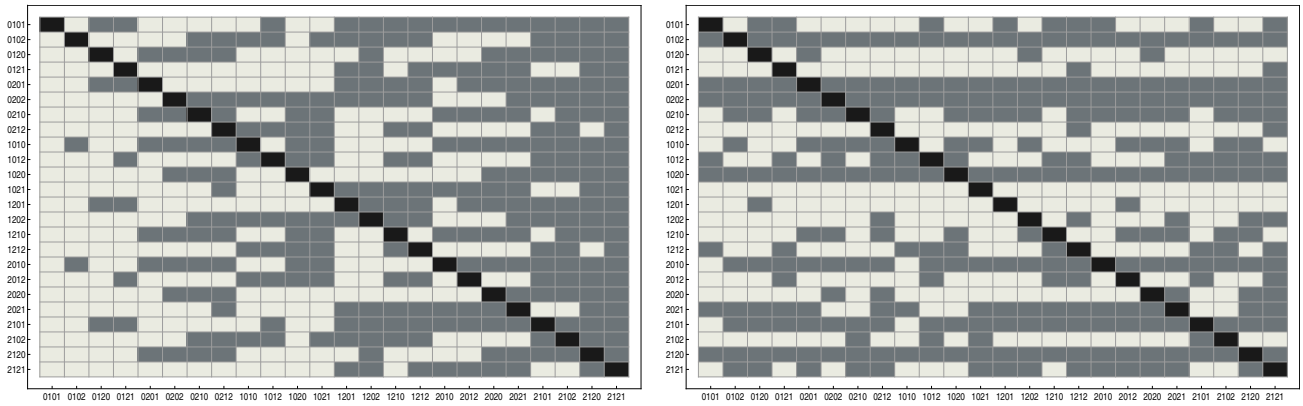
Figure 5. Structural topology of the inter-processor communication network in the Ladon module.

3.3 Low-power idling facilitated by network topology

Due to the reactive nature of sensor network applications, computation resources spend a large fraction of their time idle. This is true both for a single processing element as is the case in most existing node platforms, as well as for multi-processing platforms such as the Ladon module introduced in this paper. It is therefore important to maximize the amount of time that processing elements can spend in their lowest power sleep modes.

When employing multiple processing elements, the communication scheme implementation will have a direct impact on processor duty cycles. For example, if the implemented scheme requires processors to remain active listening for incoming communications, then, even though processing power consumption is usually a fraction of the system power consumption (since in most platforms the processor *can* be kept idle/sleeping most of the time), then the processor power dissipation would become a concern.

The mapping of the communication network topology to the hardware implementation in Ladon was specifically made to enable each processor to



(a) Shortest path routing table assuming all links have the same cost (speed).

(b) Lowest-latency routing table, taking into account the measured performance of different classes of links in the Ladon interconnect.

Figure 6. Processor routing tables for shortest path routing. Each row is the 24-bit (three-byte) complete routing table for the node whose label denotes the row. Light grey (dark grey) entries correspond to picking neighbor with lower (higher) lexicographic label, out of the two outgoing neighbors of each node. Diagonal entries (black) correspond to the path from a node to itself.

spend all of its idle time in its lowest power *stop* mode (*low power mode 4, "LPM4"*). In this mode, all the processor's subsystems are shutdown, including its clock generation modules, and it can only be woken up from sleep by an external stimulus. It can however resume active operation from this mode in less than 1μs, and can thus be automatically woken up *upon* the initiation of incoming communications.

The goal of lowest-power idling is thus achieved by mapping each directional edge in the network topology to an SPI master-slave pair. Each SPI master generates the clock for the SPI slave, thus the communication can still occur even though the slave has its clock generators disabled and is in deep sleep mode [26]. For the SPI interfaces implemented in software (dashed links in Figure 5), the interfaces are mapped to I/O pins which are interrupt capable, thus although software driven, can also be used for destination wakeup.

3.4 Unicast and broadcast communication

To support the execution of applications and systems software in a variety of programming models, it is necessary to be able to achieve high speed communications between the processing elements. One aspect of this is the speed achieved on a single link (in our case, on a single SPI master to slave connection). Another aspect is the performance of communication across non-adjacent processors, and hence the performance of routing of data across multiple hops.

Multiple algorithms exist for dynamically determining the next-hop in forwarding data along a multi-hop path in Kautz networks, including *long-path* or *greedy* routing, and *shortest-path* routing. Both of these algorithms work based on the structure that exists in node labels in Kautz networks, and are described in detail in the literature [17]. Despite resulting in a larger number of hops than the minimum (achieved by shortest-path routing), long-path routing may still be of interest in some situations, as it distributes traffic more evenly across the interconnect.

For small network sizes, there is little incentive to dynamically compute the address of the next hop, since, for low degree networks, the routing table for reaching *all* destinations can be represented very compactly. For the 24-node topology employed in Ladon, a table defining the next hop to *all* other nodes is only 3 bytes — 24 one-bit entries indicating which of the node's two outgoing edges yields the shortest path next-hop for a given final destination.

The computed shortest path routing tables for all nodes in the Ladon module's interconnect topology is shown in Figure 6(a), when assuming that all links have the same performance. When accounting for the difference in speed between hardware- and software-driven SPI links which will be shown in Section 5, the lowest-cost routing table obtained is that shown in Figure 6(b). In Figure 6,

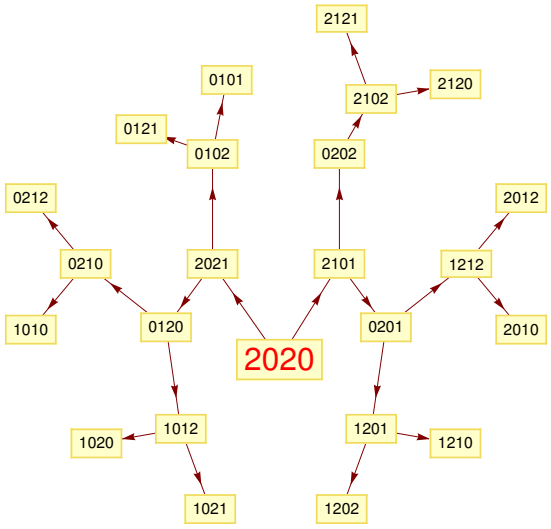


Figure 7. Logical topology of the shortest-path spanning tree for broadcasts from the Ladon module interface connector, via processor with label 2020, to all 24 processors.

each row corresponds to the 24 1-bit entries for a given node. Entries along the diagonal (black), correspond to the path between a node and itself (no routing necessary). A 0 or light-grey box (1 or dark-grey box) corresponds to the indication of the lexicographically smaller (larger) outgoing node label as the next hop on the shortest path to the given destination. For example, greedy label-based routing between 0121 and 2101 employs the path 0121→1212→2121→1210→2101, a four-hop path. In contrast, the shortest path route, which can be obtained from the 3-byte routing table, is 0121 → 1210[5] → 2101 — only two hops.

Broadcasts are an important communication pattern that may result from a variety of programming models mapped to the Ladon processor module. They are also important in system configuration, e.g., in loading code updates to all processors.

Efficient broadcasts can be achieved by sending data along the shortest path spanning tree rooted at the broadcast initiator. For broadcasts of data coming into the module to all nodes in the architecture, the entry point is the node with label 2020. The logical spanning tree rooted at node 2020, computed by Dijkstra's algorithm and shown in Figure 7, does not take into account the directed nature of edges in the hardware implementation (SPI master-slave relationships in the implementation). It can however be implemented by using the aforementioned shortest-path routing to achieve the logical edges shown in Figure 7. To maintain the restricted distribution implied by the spanning tree, broadcast messages contain the address of the logical "next hop" in the broadcast. Nodes receiving a broadcast (except for the twelve leaf nodes in Figure 7) forward incoming data along the shortest path to the logical next hop, at which the content of the message is processed, and the process repeated; the recursion bottoms out at the leaf nodes.

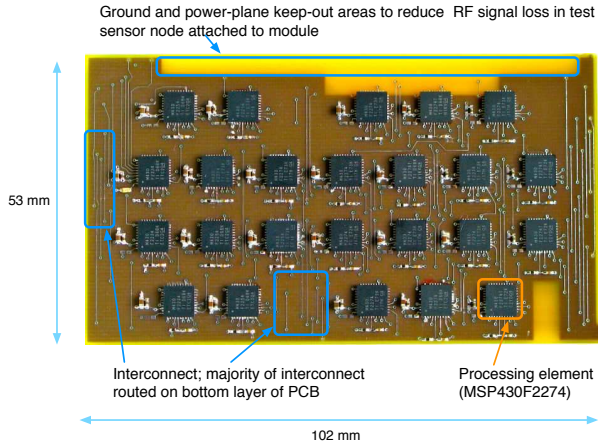
[5]Since the corresponding entry in the table (Figure 6) at position (0121, 2101) is 0 (light colored), the next hop to 2101 from 0121 (which has neighbors 1210 and 1212), is the neighbor with lexicographically smaller label, 1210.

4 System implementation

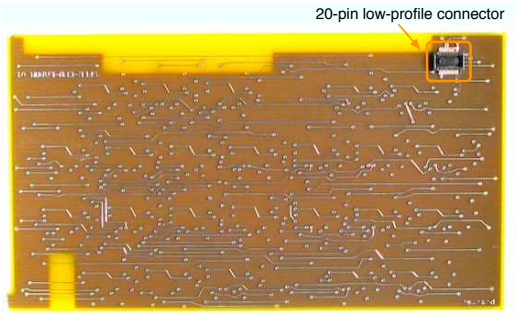
The architecture described in the foregoing sections was implemented in a four-layer printed circuit board (PCB) process, with two signal layers for interconnect routing, and two plane layers for power and ground / shielding. In our initial implementation, processing elements are only placed on one side of the PCB, to simplify and reduce costs in the board assembly process. Pictures of a prototype system are shown in Figure 8. To facilitate the use of the design as a reusable platform in actual deployments, it was desired to achieve an implementation as small in size as possible. This had influences in the choice of processing element, as well as processor package, as previously mentioned in Section 3.2. (Similar processor package size choice motivations have been reported elsewhere in the recent literature [28].)

Communication and power interfaces to the module are implemented using a low-profile 20-pin connector. The interface (Figure 9) provides two power rails (3.0 V and 1.8 V, the latter of which is currently not used), a bi-directional data interface, and an interface for debugging. Each processor in the system must currently be programmed with a basic bootloader (and its node-specific three-byte routing table), after which the system as a whole can be programmed using the normal data communication interface.

The fully-assembled 24-processor module size is 53×102×3 mm, and with the use of more printed circuit board layers, e.g., migrating to a six-layer PCB, and double-sided population of processors on the PCB, the area can easily be halved. Other directions for further reducing the size of the implementation (other than the use of smaller-sized networks, or the



(a) Top layer of PCB, with the 24 MSP430F2274 microcontrollers, and portions of the interconnect.



(b) Bottom layer of PCB, with most of the interconnect, as well as the communication and power interface connector.

Figure 8. Images of the Ladon processor module implementation.

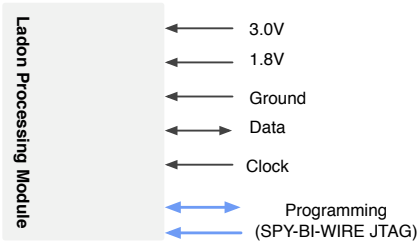


Figure 9. Power, communication and debugging interface specification for the Ladon module connector.

Table 3. Component cost breakdown for Ladon processing module.

Component	Cost (USD)
Processing elements (TI MSP430F2274)	3.35×24
Passives (resistors, capacitors)	0.10×24
Interface connector	0.54×1
Printed circuit board	1.00×1
Total	84.34

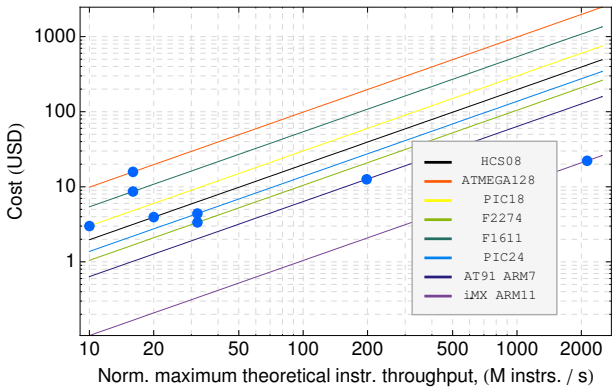


Figure 10. Trends in peak performance versus cost for a selection of contemporary microcontrollers. The overlaid lines correspond to the extrapolated cost-performance points that would be attained when employing multiple processing elements of a given type. Note the different rankings of the architectures in terms of price-performance, as compared with Figure 1 (idle-power versus performance).

design of custom integrated circuits), is the migration to chip-scale packaging on a PCB, or to more advanced packaging techniques such as *system-in-package (SiP), package-on-package (PoP)* or *redistributed chip package (RCP)* [6] technologies.

Table 3 provides a summary of the system costs — the largest costs arise from the microcontrollers (TI MSP430F2274). Due to the nature of the network topology employed for the interconnect, smaller and larger networks can be constructed with the same node degree (number of outgoing links per processor). This directly impacts scalability of constructing smaller or larger incarnations of the Ladon module, as an identical per-node mapping of communication interfaces can be employed. Figure 10 shows the trend in cost versus theoretical maximum in-

struction throughput for several commercial embedded processing elements. The *actual* achievable performance will naturally vary, and Section 5 provides a preliminary study of the performance achievable with the Ladon module. We expect experience with the platform, and the evolution of its system software to lead to better evaluations and observed performance with time.

5 System Evaluation

Several metrics are of interest in evaluating the performance of the Ladon module. On one hand, the system's minimum power consumption when all processing elements are idle determines whether the goal of low idle power in comparison to a high performance processor was met. On the other hand, the peak performance of the system determines the utility of the system in actually achieving the goal of a performance-scalable architecture.

When inactive, all processing elements in the Ladon module are in their lowest power mode, LPM4 on the MSP430F2274, and they dissipate as little as 0.1 μW each. Measurement of this sleep power consumption for the MSP430F2274 has been presented several times in the literature [25], and is not repeated here. Since the architecture takes advantage of hardware and software resources built into the MSP430F2274 to implement the communication topology, there are no additional integrated circuits in the hardware outside the processors, and thus no additional source of power dissipation. Comparing this idle power dissipation to those of contemporary higher-performance microcontrollers, the *entire* Ladon module with 24 processing elements has an idle power dissipation *over an order of magnitude smaller* than a low power ARM 7 (AT91SAM7S256), and over two orders of magnitude smaller than the lowest power mode of a low-power ARM 11 implementation (i.MX31), details of which are listed in Table 1.

The dynamic power consumption, as well as the performance achieved from the system, is dependent on the application(s) being executed on the platform, as well as the system software (such as any operating system) being employed. To study the performance of the system in a more independent manner, we study the system's performance for a number of important primitives that are independent of application and operating system, but provide insight into the system's performance:

- Interconnect performance, in terms of the jitter and signal distortion witnessed on the interconnect lines, at various communication speeds.
- Communication performance (bits per second) achieved with the hardware-driven as well as the software-driven communication interfaces.

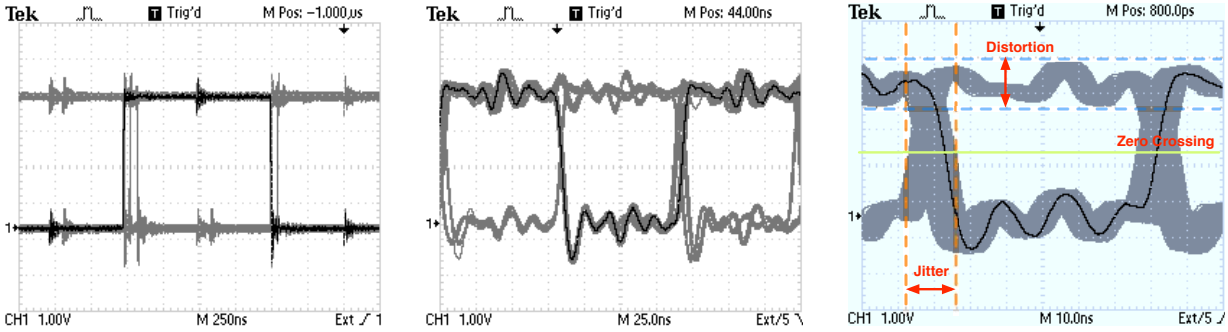
One technique to characterize the signal performance of an interconnect is the use of an *eye diagram*. Eye diagrams display a large collection of measurements of the timing and signal quality properties of low-to-high and high-to-low signal transitions on an interconnect, on a single plot. They can be used to study properties of the interconnect such as jitter (fluctuations in time of arrival of data relative to a time reference), data throughput, as well as signal distortions. An eye diagram for an ideal interconnect is a rectangle, whose left and right edges represent the quality of rising and falling signal edges, and whose top and bottom represent logic 1 and logic 0 signal levels. The width of the box corresponds to a bit period. In real systems, the width of the left and right edges denote signal jitter, and the fluctuations in the top and bottom indicate signal distortion. In general, a more "open" eye is preferred to a "closed" eye.

Figure 11 shows the eye diagrams measured on the hardware, for a representative interconnect link in the Ladon module, driven by the hardware SPI interface. The hardware implementation permits speeds of up to 8 Mb/s, with minimal jitter (14 ns) as well as minimal signal distortion (about 1 V peak-to-peak), as can be seen in Figure 11(d). The signal distortion is small enough to clearly differentiate high and low logic levels (i.e., the eye is "open"), but may further be improved by adding termination resistors to the interconnect traces in the hardware implementation; this is one direction of our ongoing work.

For software driven links, the maximum speed we achieve from our current implementation is 1 Mb/s. The electrical properties of the software-driven links are identical to those of the hardware-driven interfaces, and Figure 12 shows the eye-diagram for a software-driven link at its maximum speed.

6 Summary and Future Directions

This paper presented the motivation, architecture and implementation of Ladon, a 24-processor module for use in wireless sensor network platforms. Ladon achieves both ultra-low idle power dissipation (under 10μW), as well as the potential



(a) Eye diagram at 307 kb/s (b) Eye diagram at 5.7 Mb/s (c) Eye diagram at 8.0 Mb/s

Figure 11. Eye diagrams showing the signaling performance on the printed circuit board. Performance of up to 8 Mb/s is achieved with minimal jitter (14 ns). The observed "ringing" in the signal is due to the choice not to employ signal termination for each communication link. It does not lead to a "closing" of the eye, and thus the choice was an acceptable price/design complexity-performance tradeoff.

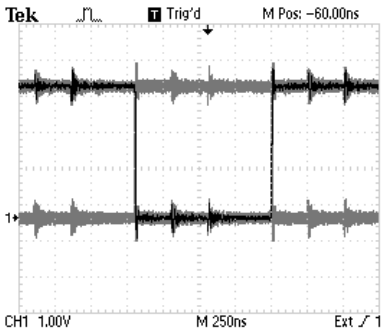


Figure 12. Eye diagram showing the signaling performance for software-driven SPI links. The maximum data rate we achieve using software-driven SPI is 1 Mb/s.

for computation throughput as high as 384 M instructions per second. The processors in the system are interconnected in a high-performance network topology in the form of a Kautz network. The topology is mapped to a combination of hardware- and software-driven serial peripheral interface (SPI) communication links, with measured speeds of up to 8 Mb/s.

In addition to the low-level system software that supports applications executing on Ladon modules (bootloaders and packet forwarding handlers), we are investigating the development of more comprehensive system software to better take advantage of hardware resources. One possible direction is the use of existing systems such as [8], and our experience writing applications over our current low-level system support infrastructure will likely provide further insights into appropriate system software directions for hardware platforms built using Ladon.

References

[1] R. Adler, M. Flanigan, J. Huang, R. Kling, N. Kushalnagar, L. Nachman, C.-Y. Wan, and M. Yarvis. Intel mote 2: an advanced platform for demanding sensor network applications. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 298–298, New York, NY, USA, 2005. ACM Press.

[2] A. Y. Benbasat and J. A. Paradiso. A compact modular wireless sensor platform. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 56, Piscataway, NJ, USA, 2005. IEEE Press.

[3] J. Beutel, O. Kasten, and M. Ringwald. Poster abstract: Btnodes – a distributed platform for sensor nodes. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 292–293, New York, NY, USA, 2003. ACM Press.

[4] N. Edmonds, D. Stark, and J. Davis. Mass: modular architecture for sensor systems. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 53, Piscataway, NJ, USA, 2005. IEEE Press.

[5] V. Ekanayake, I. Clinton Kelly, and R. Manohar. An ultra low-power processor for sensor networks. In *ASPLOS-XI: Proceedings of the 11th international conference on Architectural support for programming*

languages and operating systems, pages 27–36, New York, NY, USA, 2004. ACM Press.

[6] Freescale Semiconductor Inc. Semiconductor Packaging Technologies: System-in-Package, Package-on-Package and Redistributed Chip Packaging: Progressing Toward 3G Radio-in-Package. 2007.

[7] T. Hammel and M. Rich. A higher capability sensor node platform suitable for demanding applications. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 138–147, New York, NY, USA, 2007. ACM.

[8] C.-C. Han, M. Goraczko, J. Helander, J. Liu, B. Priyantha, and F. Zhao. Comos: An operating system for heterogeneous multi-processor sensor devices. Technical Report MSR-TR-2006-177, Microsoft Research, December 2006.

[9] D. Harel and Y. Koren. Graph drawing by high-dimensional embedding. *Proceedings of 10th Int. Symp. Graph Drawing (GD'02), Lecture Notes in Computer Science*, 2528:207–219, 2002.

[10] M. Hempstead, N. Tripathi, P. Mauro, G.-Y. Wei, and D. Brooks. An ultra low power system architecture for sensor network applications. In *ISCA '05: Proceedings of the 32nd annual international symposium on Computer Architecture*, pages 208–219, Washington, DC, USA, 2005. IEEE Computer Society.

[11] J. L. Hill and D. E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.

[12] W. H. Kautz. Bounds on directed (d,k) graphs. *Theory of cellular logic networks and machines*, AFCRL-68-0668 Final report:20–28, 1968.

[13] D. Lymberopoulos, N. B. Priyantha, and F. Zhao. mplatform: a reconfigurable architecture and efficient data sharing mechanism for modular sensor nodes. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 128–137, New York, NY, USA, 2007. ACM.

[14] D. Lymberopoulos and A. Savvides. Xyz: a motion-enabled, power aware sensor node platform for distributed sensor network applications. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 63, Piscataway, NJ, USA, 2005. IEEE Press.

[15] D. McIntire, K. Ho, B. Yip, A. Singh, W. Wu, and W. J. Kaiser. The low power energy aware processing (leap)embedded networked sensor system. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 449–457, New York, NY, USA, 2006. ACM Press.

[16] L. Nachman, R. Kling, R. Adler, J. Huang, and V. Hummel. The intel mote platform: a bluetooth-based sensor network for industrial monitoring. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 61, Piscataway, NJ, USA, 2005. IEEE Press.

[17] G. Panchapakesan and A. Sengupta. On a lightwave network topology using kautz digraphs. *IEEE Trans. Comput.*, 48(10):1131–1138, 1999.

[18] Parallax Inc. Propeller™ P8X32A Preliminary Datasheet . 2007.

[19] C. Park, J. Liu, and P. H. Chou. Eco: an ultra-compact low-power wireless sensor node for real-time motion monitoring. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, number 54, Piscataway, NJ, USA, 2005. IEEE Press.

[20] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 48, Piscataway, NJ, USA, 2005. IEEE Press.

[21] J. Portilla, A. de Castro, E. de la Torre, and T. Riesgo. A modular architecture for nodes in wireless sensor networks. 12(3):328–339, 2006.

[22] J. Rolim, P. Tvrdík, J. Trdlička, and I. Vrt'o. Bisecting de bruijn and kautz graphs. *Discrete Appl. Math.*, 85(1):87–97, 1998.

[23] P. Salinger and P. Tvrdík. All-to-all scatter in kautz networks. In *Euro-Par '98: Proceedings of the 4th International Euro-Par Conference on Parallel Processing*, pages 1057–1061, London, UK, 1998. Springer-Verlag.

[24] B. Schott, M. Bajura, J. Czarnaski, J. Flidr, T. Tho, and L. Wang. A modular power-aware microsensor with $>$ 1000x dynamic power range. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 66, Piscataway, NJ, USA, 2005. IEEE Press.

[25] Texas Instruments, Inc. Datasheet, MSP430x22x2, MSP430x22x4 Mixed Signal Microcontroller. 2006.

[26] Texas Instruments, Inc. Datasheet, MSP430x2xx Family User's Guide. 2006.

[27] B. Warneke and K. J. Pister. An ultra-low energy microcontroller for Smart Dust wireless sensor networks. In *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC 2004, IEEE International*, pages 316–317, February 2004.

[28] S. Yamashita, T. Shimura, K. Aiki, K. Ara, Y. Ogata, I. Shimokawa, T. Tanaka, H. Kuriyama, K. Shimada, and K. Yano. A 15×15 mm, $1\,\mu$A, reliable sensornet module: enabling application-specific nodes. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 383–390, New York, NY, USA, 2006. ACM Press.