MBus: An Ultra-Low Power Interconnect Bus for Next Generation Nanopower Systems

Pat Pannuto, Yoonmyung Lee, Ye-Sheng Kuo, ZhiYoong Foo, Benjamin Kempke, Gyouho Kim, Ronald G. Dreslinski, David Blaauw, and Prabal Dutta
Electrical Engineering and Computer Science Department
University of Michigan, Ann Arbor, MI 48109
{ppannuto,sori,samkuo,zhiyoong,bpkempke,gyouho,rdreslin,blaauw,prabal}@umich.edu

Abstract

As we show in this paper, I/O has become the limiting factor in scaling down size and power toward the goal of invisible computing. Achieving this goal will require composing optimized and specialized—yet reusable—components with an interconnect that permits tiny, ultra-low power systems. In contrast to today's interconnects which are limited by power-hungry pull-ups or high-overhead chip-select lines, our approach provides a superset of common bus features but at lower power, with fixed area and pin count, using fully synthesizable logic, and with surprisingly low protocol overhead.

*We present **MBus**, a new 4-pin, 22.6 pJ/bit/chip chip-to-chip interconnect made of two "shoot-through" rings. MBus facilitates ultra-low power system operation by implementing automatic power-gating of each chip in the system, easing the integration of active, inactive, and activating circuits on a single die. In addition, we introduce a new bus primitive: power oblivious communication, which guarantees message reception regardless of the recipient's power state when a message is sent. This disentangles power management from communication, greatly simplifying the creation of viable, modular, and heterogeneous systems that operate on the order of nanowatts.*

To evaluate the viability, power, performance, overhead, and scalability of our design, we build both hardware and software implementations of MBus and show its seamless operation across two FPGAs and twelve custom chips from three different semiconductor processes. A three-chip, 2.2 mm^3 MBus system draws 8 nW of total system standby power and uses only 22.6 pJ/bit/chip for communication. This is the lowest power for any system bus with MBus's feature set.

ISCA'15, June 13-17, 2015, Portland, OR USA
ACM 978-1-4503-3402-0/15/06.
http://dx.doi.org/10.1145/2749469.2750376

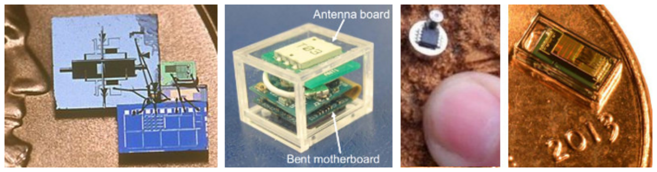


Figure 1: Recent micro-scale systems. From left to right Pister's Smart Dust [37], the 1cc computer [21], "Smart Dew" [31], and our system with MBus show the feasibility of micro-scale systems, the techniques, such as 3D-stacking, employed to realize their size, and the volume and surface area constraints that lead to their extremely limited energy storage, energy harvesting, and I/O capabilities.

1. Introduction

For nearly three decades, most microcontrollers have come with the same peripheral interfaces: SPI, I^{2}C, and UART. In this paper, we argue that embedded microcontroller technology has now progressed in terms of energy consumption and miniaturization to the point where existing interfaces no longer meet the needs of these emerging systems. We show fundamental drawbacks in the area requirements and energy consumption of all existing embedded interfaces that demand the design of a new interface if modular, embedded hardware is to scale beyond today's centimeter-scale devices.

This paper focuses on embedded systems and the greater ecosystem of application-driven hardware. These are systems that require *assembly-time* modularity. They compose sensing, computation, communication, and other novel functions from a collection of hardware building blocks to realize new application-specific devices. The burgeoning "Internet of Things" is a product of this ecosystem.

Traditional system design has scaled as small as centimeter-sized "wearable" devices. The research community has pushed the envelope further, exploring an array of "micro-scale"—ultra-low power and millimeter-sized—building blocks including processors [8, 37], radios [4, 7, 26], other long-range communication technologies [6], ADCs [28], regulators [33], timers [15, 16], and sensing frontends [9, 17]. However, few of these components have been integrated into complete systems. Some projects, like those shown in Figure 1, have built complete systems, but these usually have been monolithic, similar to early computers, designed with tight integration using custom interfaces [21, 29, 30, 31, 36].

We claim that introducing modularity and reusability to micro-scale systems will more rapidly yield the next generation of intelligent devices. Our aim is to facilitate an ecosystem of micro-scale embedded systems. We have created many of the building blocks of this ecosystem—processors, sensors, storage, and communication—but find that the greatest impediment to working, modular, micro-scale systems is no longer the building blocks themselves, but the resources demanded by I/O, the glue that binds them together.

To address this problem, we break from conventional interconnect buses that employ power-hungry pull-ups in open-collector configurations or I/O-expensive chip-select lines. We propose MBus, a pair of "shoot-through" rings—one `CLK` and one `DATA`—and a clean-slate bus design for the emerging class of micro-scale systems. We realize a design with features that are a superset of today's interconnects but at lower power (22.6 pJ/bit/chip), with fixed area and pin count (4), using fully synthesizable logic, with minimal protocol overhead ($19-43$ cycles), and no local clock generation.

The MBus design transparently supports a number of energy conservation mechanisms that are required for ultra-low power operation, but which raise additional system-level challenges. For example, to minimize static leakage, systems power-gate sub-circuits or whole chips when they are not in use. But this requires one subsystem to wake up another subsystem before communications can occur, often utilizing a custom method or protocol, like the wakeup sequence in Lee's I^2C variant [14]. In practice, these schemes require every sender to either know the power state of every recipient in advance or to send a wakeup sequence before every message. This design also requires that each chip have the ability to self-start: to stably go from the initial application of power to a start state in a glitch-free manner with no external clock or reset signal. Requiring such wakeup circuitry on every chip adds design overhead to every chip in a micro-scale system.

MBus solves the wakeup problem by providing the abstraction of an always-on recipient: a sender may send a message to any recipient, regardless of the recipient's power state, and the recipient will receive the message. MBus enables this "power oblivious communication" abstraction by taking on the power-management burden for each chip, allowing MBus to wake the chip when necessary. This bus-provided wakeup eliminates the need for custom wakeup circuitry, often derived from semiconductor process-specific circuits such as delay chains, simplifying the design of the entire chip.

We analyze the MBus protocol and several physical implementations to show that it is an efficient and effective bus for micro-scale systems. Its message length-independent overhead supports a wider range of messages, from bytes to hundreds of kilobytes, efficiently; its ring topology, with only FETs driving gates, results in an energy-efficient 22.6 pJ/bit/chip measured on real hardware; and its optional, lightweight enumeration provides an area-free efficient multiplexing of a limited address space. The power-oblivious principle enables the seamless integration of traditional and ultra-constrained modules. This allows for re-use across classes of technology and provides current systems access to efficient, micro-scale building blocks. We close with design caveats of MBus and a discussion of future directions for the protocol.

2. Related Work and the Case for a New Bus

In this section, we show why existing bus protocols like I^2C, CAN, SPI, and I^2S are not viable system interconnects for modular, resource-constrained systems due to their energy use, protocol overhead, pin count, and system design requirements.

2.1. I^2C, 1-Wire, CAN, and Other Open-Collector Buses

Many interconnects are built on an open-collector or open-drain design (e.g. I^2C [23], SMBUS [1], and CAN [11]). This circuit construct turns each bus line into a wired-AND; one or many devices can drive a `0` on the bus, but if nothing actively drives low, then pull-up resistors pull each line high. The advantages of this approach are decentralized arbitration and multi-tiered priority. The pull-up resistors, however, are not energy efficient and result in designs that have up to three orders of magnitude worse energy per bit than MBus.

To illustrate, consider an idealized I^2C configuration running at 1.2 V that we try to optimize for energy consumption. I^2C typically requires the pull-up resistor be sized to accommodate 400 pF of total bus capacitance, but let us relax that to 50 pF for micro-scale systems; fast mode I^2C has a 400 kHz clock and must reach 80% V_{DD} in 300 ns, but let us relax that (eliminate setup and hold time) to the full half-cycle (1.25 µs). This relaxed I^2C bus requires a pull-up resistor no greater than 15.5 kΩ. To generate the bus clock, this resistor is shorted to ground for a half period, dumping the charge in the bus wires, pads, and FET gates (23 pJ) and dissipating power in the resistor (116 pJ). The clock line then floats for a half cycle and the resistor pulls it high (35 pJ). Thus, generating the clock alone draws 69.6 µW. Eliminating the switching power – the 23 pJ/bit charging and discharging of the wire, pad, and gate capacitance – requires complex adiabatic clocks, outside the scope of our design [27, 38]. MBus finds its energy gains by eliminating the 151 pJ/bit lost to the pull-up resistor.

2.2. I^2C Variations

One conceivable idea that we briefly explore for reducing the impact of the pull-up might be to "unbalance" the clock as shown in Figure 2. This would allow the designer to nearly double the size of the pull-up resistor (halving the power draw) while maintaining the same bus clock period and minimizing the impact of the SCL line on energy usage. Unfortunately, this concept does not reduce the energy consumed by the pull-up while pulling up, nor does it reduce the energy consumed by the data line when transmitting `0`'s. Unbalanced clocks would also require local timing modifications, costly in energy and complexity, ruling out this possibility.

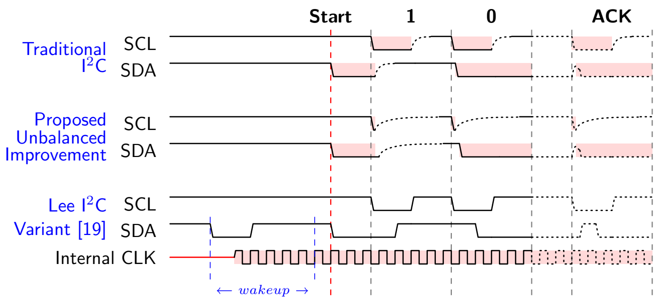


Figure 2: Waveforms of I^{2}C and variants. A comparison of traditional I^{2}C and some of the proposed variations. Shaded areas are power-expensive protocol elements.

Lee et al. reduce I^{2}C power draw by designing an "I^{2}C-like" bus that replaces the pull-up resistor with logic that actively pulls the bus high and a low-energy "bus keeper" circuit that preserves the last value [14] (similar to I^{2}C Ultra Fast-mode [23]). While this approach eliminates the pull-up, it does so at the cost of requiring a local clock running $5\times$ faster than the bus clock, the energy inefficiency seen in Figure 2. We cannot see a means for designing an I^{2}C or I^{2}C-like bus without either a pull-up or a fast-running internal clock. While the internal clock is not as energy-intensive—Lee's system is able to reduce bus energy to 88 pJ/bit (4 times that of MBus)—, Lee's design also requires hand-tuned, process-specific ratioed logic; requiring manual tuning of every chip runs counter to the goals of a general-purpose bus.

Pannuto found that while Lee's I^{2}C variant is designed with commercial interoperability in mind, in practice actual interoperation required an FPGA to translate between I^{2}C and the "I^{2}C-like" bus [24]. MBus eschews the "partial compatibility" that I^{2}C-like buses provide and uses the clean break to reconsider the primitives provided by the system interface, allowing the addition of features such as power oblivious communication, broadcast messages, and efficient transaction-level acknowledgments.

2.3. SPI, I^{2}S, Microwire, and Other Single-Ended Buses

As single-ended buses, SPI and its derivatives do not suffer from the power challenges faced by open-collectors and have little to no protocol overhead. SPI, however, requires a unique chip-select line for every slave device. In a modular system with a variable (and unknown until *system design* time) number of components, it is difficult to choose the "right" number of chip select lines a priori—too few impede modularity and too many violate the area constraints of micro-scale systems. Additionally, SPI requires a single master that coordinates and controls access to all slave devices, and it requires all communication between slave devices to go through the master node. This more than doubles the communication cost for slave-to-slave transmissions: every message is sent twice plus the energy of running the central controller. A further subtle, yet critical, implication of a single-master design is that all communication is master-initiated. For a sensor to signal a microcontroller (i.e. an interrupt), it requires an additional I/O line, a resource that is unavailable to micro-scale systems.

Alternative configurations such as daisy-chained SPI can eliminate the chip-select overhead but do not solve the multi-master/interrupt issue and require the addition of a protocol layer to establish message validity. As a system-wide shift register, a daisy-chain configuration adds overhead proportional to both the number of devices and the size of the buffer in each device. SPI and its derivatives are fundamentally incompatible with size-constrained microsystems.

2.4. Bus Designs from Other Disciplines

The original token ring protocol requires that empty frames are continuously passed so that nodes can grab the token when they need it. Low-power systems rely on low duty cycles to remain efficient. Using tokens in place of arbitration either requires occasional empty frames to pass the token, with an inherent latency/energy tradeoff, or a sacrifice of multi-master capability.

Some of the new network-on-chip (NoC) protocols that have been developed, such as Nehalem's QuickPath [12], include power-aware features like MBus. These buses seek to move large amounts of data often via wide parallel buses, within a siloed system rather than compose independently designed, modular components.

Recently, work in energy-efficient, short-reach data links has lead to energy performance as low as 0.54 pJ/bit [25]. These designs target high-performance computing applications, however, utilizing complex transmit and receive circuitry with high-speed clocks [18] and add requirements such as a common substrate with carefully carved channels [25]. As a platform bus, MBus aims to support a wider diversity of packages and physical interconnection technologies. While some components, such as an efficient charge-pump design, could be adopted, we leave their integration to future work.

2.5. Power Savings Create Communication Problems

Ultra-constrained systems need to aggressively conserve power. Devices that are left on or in standby allow communication to occur on-demand. Ultra-low power systems, however, realize their power goals in part through aggressive duty-cycling. A power-gated node must be awakened before it can communicate, presenting interoperability (how to wake a node) and run-time (when to wake a node) challenges.

Lee et al. identify this wakeup issue and modify their protocol to include a wakeup signal: an I^{2}C start bit followed shortly by a stop bit. This requires the sender to know the receiver's power state in advance or to unconditionally send the wakeup sequence before every message. Due to implementation choices, the minimum time between the start and stop bits of the wakeup sequence and the time until the chip is awake after the stop bit is received varies from chip to chip, requiring hand-tuning and conservative estimates. This design also requires a self-starting power-on circuit in each chip.

	I^2C	SPI	UART	Lee-I^2C	MBus
Critical					
I/O Pads (n nodes)	**2/4**[†]	$3 + n$	$2 \times n$	**2/4**[†]	**4**
Standby Power	**Low**	**Low**	**Low**	**Low**	**Low**
Active Power	High	**Low**	**Low**	Med	**Low**
Synthesizable	**Yes**	**Yes**	**Yes**	No	**Yes**
Global Uniq Addresses	128	—	—	128	$\mathbf{2^{24}}$
Multi-Master (Interrupt)	**Yes**	No	No	**Yes**	**Yes**
Desirable					
Broadcast Messages	No	Option	No	No	**Yes**
Data-Independent	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Power Aware	No	No	No	No	**Yes**
Hardware ACKs	**Yes**	No	No	**Yes**	**Yes**
Bits Overhead (n bytes)	$10 + n$	**2**[‡]	(2-3)[§] $\times n$	$10 + n$	**19, 43**[*]

[†] When wirebonding, a shared bus requires two pads/chip (or a much larger shared pad/trace)
[‡] Asserting and de-asserting the chip-select line
[§] Depending on the stop condition; assumes 8-bit frames and no parity
[*] Depends on whether short (more common) or long addressing is in use

Table 1: Feature Comparison Matrix. Population-independent area, ultra-low power operation, synthesizability, an area-free global namespace, and interrupt support are fundamental requirements for a micro-scale interconnect. Standby power is on the order of 100's of pW and active power ranges from 10's of μW to 10's of nW. Only MBus satisfies all of our required features.

In contrast, MBus takes over the power management of nodes, freeing designers from the burden of building complex self-start circuits. MBus guarantees delivery of messages, independent of the power state when a message is sent, eliminating requirements for distributed power state management.

3. Requirements for an Ultra-Constrained Bus

In this section, we summarize the requirements we have identified for the system interconnect of ultra-constrained systems based on our experience with several generations of building such devices. MBus is the first interconnect design that meets all of the requirements. Table 1 summarizes these requirements and compares MBus with popular current interchip buses and a recent research bus, Lee's I^2C variant [14].

Synthesizable. To facilitate widespread adoption, we require a process-agnostic solution, a block of "pure" HDL with no process-specific custom macros. Lee's I^2C variant requires process-specific tuning of custom ratioed logic, which adds cost, complexity, and risk to every implementation.

Low Wire Count. With sub-millimeter scale systems, the area cost required to place bonding pads (35–65 μm wide) on one edge or around the perimeter of a chip limits the ability of the system to scale. While advancements such as through-silicon vias (TSVs) help, many popular processes do not support them (e.g. IBM130 or TSMC65).

Address Space. A bus must provide a way of addressing each element, for example with hardware support (e.g. SPI chip-selects) or explicit addresses (e.g. I^2C). If addresses are used, they must both support a large number of possible devices (e.g. 2^{20}) and minimize overhead on each transmission.

Low Standby Power. Resource-constrained systems spend the majority of their time in standby so standby inefficiencies are magnified. Existing interconnects are well suited to this, so any new bus must draw less than 100 pW to be competitive.

Low Active Power. Micro-scale systems have extremely constrained power budgets. Our most constrained system is powered by a 0.5 μAh battery and targets a total system active power budget of < 40 μW (and idle power of 20 nW). For reference, recall that the I^2C clock alone uses 69.6 μW. While any absolute number will be system-dependent, to allow Amdahl-balanced system design, we target an upper limit of 20 μW total active power draw for the system interconnect.

Data-Independent Behavior. Protocols that use dedicated symbols to communicate special cases (such as an end-of-message indicator) require byte stuffing, which in pathological cases can double the length of a message. This affects the ability to reason about protocol performance both in energy and time, and in real-time systems can lead to violations of timing requirements or require artificially high provisioning.

Fault Tolerance. It must be impossible for the bus to enter a "locked-up" state due to any transient faults.

Interrupts / Multi-Master. To facilitate a diverse and unpredictable set of devices and system applications, any device must be able to initiate a transmission to any other device at any time. This requires either an efficient, non-polling based interrupt mechanism or a true multi-master design.

Efficient Acknowledgments. Many applications require reliable message transport. This feature may be directly supported by the bus protocol in hardware, or as an optional software feature if it can be made sufficiently low-overhead.

Power-Aware. Unlike deep sleep, which still loses energy to static leakage, a power-gated circuit loses all state. Ultra-constrained systems need to cold boot and later shut down sub-circuits without affecting active areas. Interfaces between power domains must be *isolated*, tied to a fixed value by an always-on logic gate, so that floating signals do not confuse active logic. To power on a power-gated circuit reliably and without introducing glitches, four successive edges, the *wakeup sequence*, must be produced:

1. Release Power Gate: Supply power to the circuit that is being activated
2. Release Clock: (Optional) After a clock generator is powered on, it requires time to stabilize before driving logic
3. Release Isolation: Outputs of a power-gated block float when off and must be isolated until they are stable
4. Release Reset: Once stable, the circuit may leave the reset state and begin interacting with the rest of the system

This sequence is fundamental to powering on sub-circuits.

Aggressively low power designs have no clock sources in their lowest-power state and no means to generate these signals. In current systems, every design requires a custom "wakeup" circuit to generate these edges, adding cost and complexity.

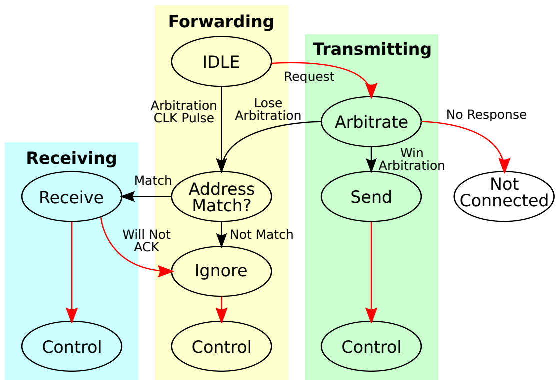


Figure 3: High-level behavior of MBus. A transaction begins when one or more nodes elect to transmit by starting an arbitration phase (4.3). The winner then transmits a destination address (4.6, 4.7), sends payload data (4.8), and interjects to end the transaction (4.9). Black arrows are synchronous, transitions that match MBus CLK edges, while red arrows are asynchronous. By leveraging the interjection primitive (4.9), MBus transmitters can reliably signal the end of message without requiring an embedded length or an out-of-band—extra wire—signal. Not shown are arrows from any state to control due to interjections and from control back to idle.

We argue that these low-level details should be hidden from the application developer. The wakeup sequence provides a clean interface to power on a system and the system interconnect should provide a clean abstraction for sending messages, i.e. one that ensures receipt independent of the target device type or immediate power state.

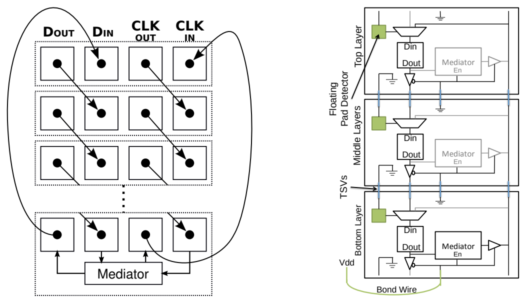
Interoperability. Not all systems may be severely resource or energy constrained, yet they may still wish to use chips designed for ultra-low power applications. Any interconnect must bridge the gap between power-conscious and power-oblivious—no notion of power-gating and no specialized constructs to support it—devices to avoid fracturing the component ecosystem and to enable reuse across all device classes.

4. MBus Design

In this section we describe the MBus design. Working from the constraints identified in Section 3, we build up MBus, discussing key design decisions and trade-offs along the way. Figure 3 presents an overview of the MBus protocol by considering the possible states for nodes during a transaction.

4.1. MBus Topology: Rings and Selectively Sharing Wires

To meet the wire count requirement, the bus topology must be independent of the number of nodes, i.e. adding another node cannot require adding a new wire. As many nodes thus share the same wires, MBus requires a scheme to avoid conflicts, driving the same line high and low. Some form of token-passing or leader-based protocol violates the efficient interrupt requirement, requiring the leader to poll to find the interrupter. MBus prevents conflicts by using two rings, CLK and DATA, as shown in Figure 4. To minimize active power, MBus clocks



(a) 3D Stack Example **(b) TSV Example**[†]

[†]Only DATA shown for clarity, the CLK line uses identical selection circuitry.

Figure 4: MBus Physical Topology. An MBus system consists of a mediator node and one or more member nodes connected in two "shoot-through" rings, one CLK and one DATA. The ring topology adapts to many system synthesis methods, including stepped 3D-stacking with wirebonding (a) and TSVs (b).

all bus logic off of the bus clock itself. This obviates the need for a local oscillator on each node. With no local clock, the rings are "shoot-through": signals pass through only a minimal amount of combinational logic from one node to the next.

4.2. Clock Generation and Bus Mediation

MBus introduces one special node, the *mediator*. The mediator is responsible for generating the MBus clock and resolving arbitration. Every MBus system must have exactly one mediator, either attached to a core device (e.g. a microcontroller device) or as a standalone component (similar to the pull-up resistors in I^2C). For ultra-low power designs, MBus power-gates all but the forwarding drivers (Wire Controller) and a minimalist wakeup frontend (Sleep Controller). The mediator must therefore be capable of self-starting. In an ultra-low power design, something must have the capability to self-start; the mediator allows that self-start requirement to be contained within a single, reusable component.

4.3. Arbitration by Mediating Shoot-Through Rings

In the idle state all nodes forward high CLK and DATA signals around the rings. A node requests the bus by breaking the chain and driving its $\mathrm{DATA}_{\mathrm{OUT}}$ low. This propagates around the DATA ring until it reaches the mediator, which does not forward DATA during arbitration. The falling edge on $\mathrm{DATA}_{\mathrm{IN}}$ triggers the mediator self-start which begins toggling CLK as soon as it is active. At the first rising edge of CLK, any arbitrating nodes sample their $\mathrm{DATA}_{\mathrm{IN}}$ line. If $\mathrm{DATA}_{\mathrm{IN}}$ is high, the node has won arbitration, otherwise it has lost. This arbitration scheme introduces a topologically-dependent priority on MBus nodes. To afford physically low-priority nodes an opportunity to send low-latency messages, MBus adds a priority arbitration cycle after arbitration. The priority arbitration scheme is similar, except it is the arbitration winner that does not forward DATA and nodes pull $\mathrm{DATA}_{\mathrm{OUT}}$ high to issue a priority request. Figure 5 shows a waveform of arbitration and priority arbitration.

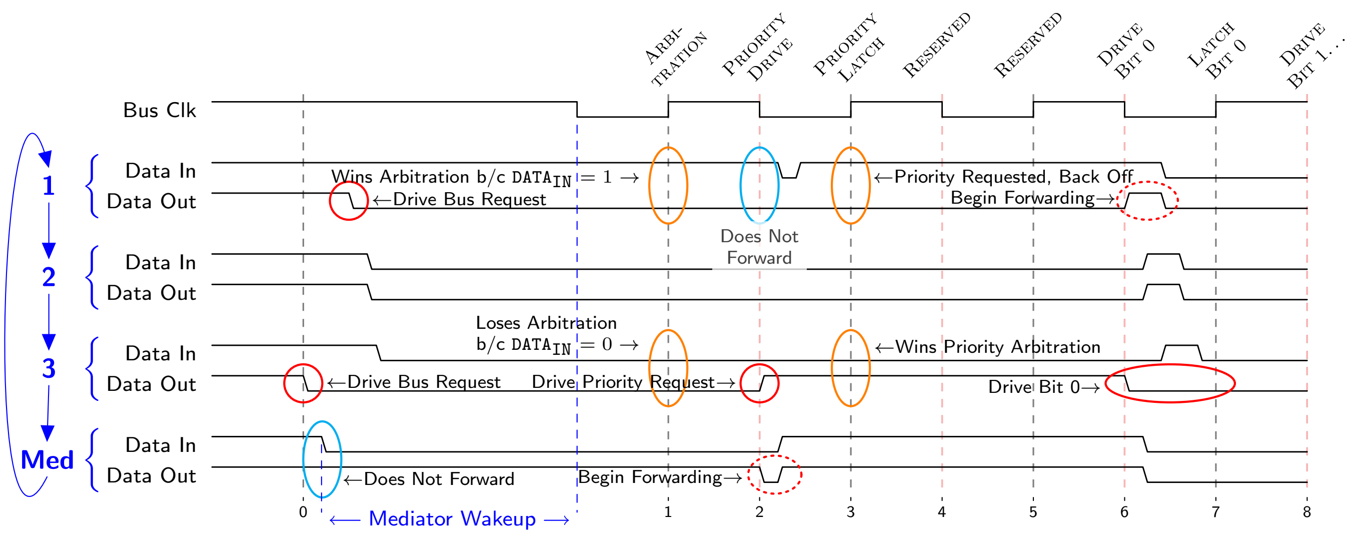


Figure 5: MBus Arbitration. To begin a transaction, one or more nodes pull down on $\texttt{DATA}_{\texttt{OUT}}$. Here we show node 1 and node 3 requesting the bus at nearly the same time (node 1 shortly after node 3). Node 1 initially wins arbitration, but node 3 uses the priority arbitration cycle to claim the bus. The propagation delay of the data line between nodes is exaggerated to show the shoot-through nature of MBus. Momentary glitches caused by nodes transitioning from driving to forwarding are resolved before the next rising clock edge.

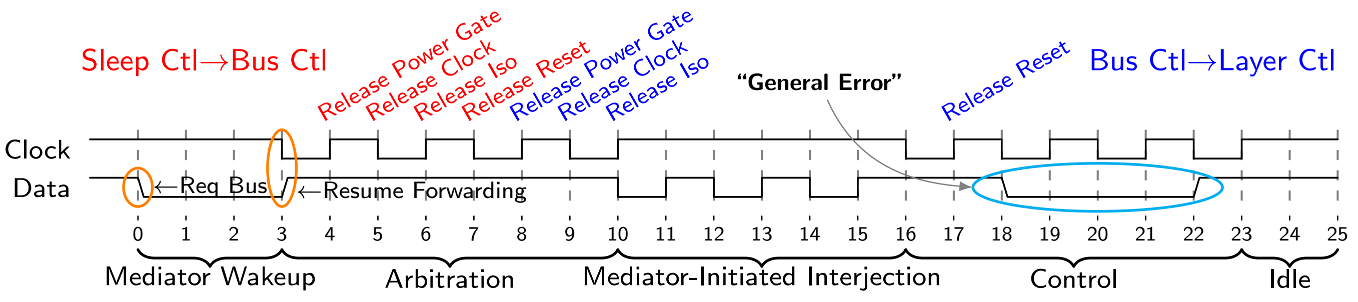


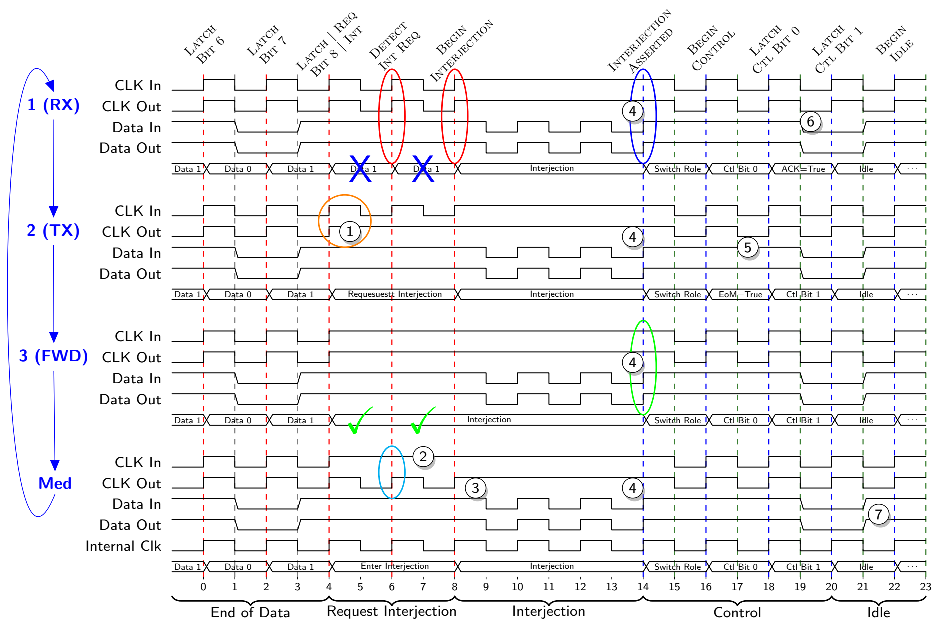
Figure 6: MBus Wakeup. Power-gated nodes repurpose the arbitration edges to wake the bus controller before data transmission starts. To self-wake, nodes can initiate a null transaction (shown here) by pulling down `DATA` and then resuming forwarding `DATA` before the arbitration edge. If no other node arbitrates, the mediator will detect no winner, raise a general error, and return the bus to idle. Arbitration produces enough edges to wake bus controllers before addressing. The null transaction produces enough edges to wake all of the MBus hierarchical power domains in a manner that is transparent to non-power-aware devices.

4.4. Transparent and Efficient Hierarchical Wakeup

For maximum power efficiency, MBus on a power-gated node leaves only a minimalist, highly-optimized frontend on continuously. To receive an MBus transmission, however, the power-gated node's bus controller must first be activated. The key insight that enables MBus's power-oblivious properties is that a power-gated node can use the edges on the `CLK` line from arbitration as the wakeup sequence (described by the Power Aware requirement of Section 3) to wake its bus controller. Because arbitration occurs before every message, all bus controllers in the ring are active by the addressing phase and can determine whether the message is destined for this node. MBus does not wake the rest of the node until an address match. This ensures that only the destination node is powered on by the receipt of a message. This design allows any node to transmit to any other node at any time while ensuring that the receiving node and only the receiving node will be powered on to receive the message.

4.5. Supporting Intra-Node Wakeups

Partially power-gated nodes will transparently wake up to receive messages, but they may also wish to voluntarily wake themselves, usually in response to a locally generated event from, say, a timer or sensor. For instance, a node with an always-on analog circuit (e.g. an ultra-low power motion detector embedded in an imager, as described in Section 6.3.2) may wish to wake the rest of the node to take a picture or send an alert message. The always-on MBus frontend provides a simple interrupt port that the component can assert. Upon interrupt request, the frontend will generate a null message, as shown in Figure 6. This null message causes the mediator to generate the clock edges needed to wake the rest of the node, such as the control circuitry, which can handle the interrupt. With this design, a power-conscious node can leverage all of its power-saving faculties without requiring support from any other system component, simultaneously maximizing interoperability and efficiency.



(a) Interjection timing with successful TX and ACK.

1. After the transmitter sends all of its data it requests interjection by not forwarding `CLK`.
2. The mediator detects that a node has stopped forwarding `CLK`.
3. The mediator stops toggling `CLK` and begins toggling `DATA` – the interjection sequence.
4. After interjection, the mediator begins clocking again. Node 1 discards the extra bits because they are not byte-aligned.
5. The transmitter signals a complete message by driving Control Bit 0 high.
6. The receiver ACK's the message by driving Control Bit 1 low.
7. After control, the mediator stops forwarding `DATA`, driving it high, and returning the bus to idle.

(b) Interjection Events.

Figure 7: MBus Interjection and Control. The MBus interjection sequence provides a reliable in-band reset signal. Any node may request that the mediator interject the bus by holding $\texttt{CLK}_{\texttt{OUT}}$ high. The mediator detects this and generates an interjection by toggling `DATA` while holding `CLK` high. An interjection is always followed by a two-cycle control sequence that defines why the interjection occurred.

4.6. Prefixes, FU-IDs, and Broadcast Messages

MBus uses an addressing scheme to direct transmissions and divides addresses into two components, a *prefix* and a *functional unit ID* (FU-ID). A prefix uniquely addresses a physical MBus interface (one of the actual chips in the system), while FU-IDs are used to address chip sub-components. FU-IDs are 4-bits, allowing for up to 16 sub-components behind each physical MBus frontend. MBus reserves prefix `0` for broadcast messages. On a shared bus, broadcast messages are cheap to implement in hardware but expensive (linear in system size) to emulate in software, motivating hardware broadcast support. MBus repurposes the FU-ID of broadcast messages as broadcast *channel* identifiers, allowing nodes to listen to only the broadcast messages they support or are interested in.

4.7. Prefix Assignment and (Optional) Enumeration

To retain the efficiency afforded by short addresses while allowing for a diverse ecosystem of unique components, MBus uses run-time enumeration to assign 4-bit *short prefixes*. Enumeration is a series of broadcast messages containing short prefixes that can be sent by any node (although in practice most likely by a microcontroller). All unassigned nodes attempt to reply with an identification message and the arbitration winner is assigned the enumerated short prefix. A result of this enumeration protocol is that a node's short prefix encodes its topological priority. Enumeration is performed once, when the system is first powered. As an optimization, devices may assign themselves a *static short prefix*, akin to I^2C addressing, so if there are no conflicts enumeration may be skipped.

Every chip design is assigned a unique, 20-bit *full prefix*. Full prefixes allow nodes to refer to one another with static addresses at the cost of 16 bits of additional overhead per message. The short prefix `0xF` is reserved to indicate full addresses, leaving MBus with 14 usable short prefixes per system. Chips may be addressed using either short or full addresses interchangeably. It is sometimes advantageous for a system to have two copies of the same chip (e.g. memory), which requires short prefixes and enumeration to disambiguate.

4.8. MBus Data Transmission and Acknowledgments

MBus transmitters drive data on the falling edge of `CLK` and receivers latch data on the rising edge of `CLK`. While standard flops can only be clocked on one edge (rising or falling), only the internal data FIFO needs to be clocked on the falling edge, thus this does not violate the synthesizability requirement. This design is modeled after the circuit shown by Kuo et. al in [13], which allows for identical setup and hold margins, easing interoperability when driving unknown loads.

At the end of a message, the receiver either ACKs or NAKs the entire message. In MBus, any node may terminate any message at any time, even a forwarder. The transmitter may end the message when it is finished, or the receiver may interject mid-message to indicate error, e.g. buffer overrun. Thus, by not interjecting, a receiver *implicitly* ACKs every byte. This is less powerful than I^2C ACKs, which can detect a dead receiver after the first byte, but is more efficient during error-free operation and allows MBus to scale to long (multi-kilobyte) messages with a fixed, length-independent overhead.

4.9. In-line Interjections End Messages, Provide Reliable Reset, and Enable Responsive Messaging

At any point, the bus may be interrupted by an MBus *interjection*. In normal MBus operation, `DATA` never toggles meaningfully without a `CLK` edge. This allows us to design a reliable, independent interjection-detection module, essentially a saturating counter clocked by `DATA` and reset by `CLK`. The interjection signal acts as a reset signal to the bus controller, clearing its current state and placing it in *control* mode. MBus control is two cycles long and is used to express why the bus was interjected, either an end-of-message that is ACK'd or NAK'd or to express some type of error. Figure 7 shows the end of an MBus transaction sent from Node 2 to Node 1 that is ACK'd. Notice that the MBus interjection request mechanism, holding `CLK` high, results in nodes observing a varying number of clock edges. MBus requires that messages be byte-aligned to resolve this potential ambiguity, potentially requiring a small amount (up to 7 bits) of padding to be added to MBus messages.

MBus interjections are used both for extreme cases, such as rescuing a hung bus or indicating receiver error, and as a regular end-of-message signal. Any node may generate an interjection at any time. This allows for unambiguous signalling of control functions that can resynchronize a bus without requiring out-of-band signals like chip-selects or a reset line. This further allows a node with a latency-sensitive message to interrupt an active transaction, enabling responsiveness across a diverse array of workloads not possible with current buses.

5. Implementation

Figure 8 shows the complete MBus Verilog design. For non-power-conscious designs, only the Bus Controller is required. To support self and system power-gating, the additional Sleep Controller, Wire Controller, and optional Interrupt Controller are added. The Layer Controller is a generic module acting as a stand-in for the rest of the node. While most nodes will likely have a local oscillator, it is possible to interface with the Bus Controller using only edges harvested from the bus clock.

Table 2 shows the cost in area for each of the MBus components (excluding I/O pads) when synthesized for an industrial 180 nm process, with comparisons to SPI, I^{2}C, and Lee's I^{2}C variant. MBus imposes an area cost penalty, but offsets this with its additional features.

We implement MBus in twelve chips in three different technologies (65, 130, and 180 nm CMOS) and two FPGA fabrics (Actel SmartFusion [3] and Microsemi IGLOO nano [19]) and find that all interoperate without error and without tuning.

6. Evaluation

To evaluate MBus, we first evaluate the protocol, then perform an in-depth analysis on its energy performance. Next we build and analyze two representative micro-scale systems. Finally, we consider the scalability and interoperability of MBus.

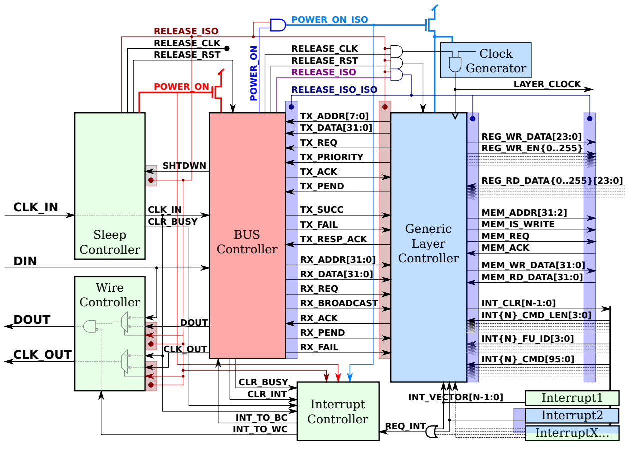


Figure 8: MBus Implementation. We implement MBus as a series of composable Verilog modules. The module coloring represents the three hierarchical power domains: green modules (Sleep, Wire, and Interrupt Controllers) are always powered on, red modules (Bus Controller) are powered during MBus transactions, and blue modules (Layer Controller, Local Clock) are powered only when the node is active. Critically, MBus itself requires no local oscillator. The generic layer controller provides a simple register/memory interface for a node, but its design is not specific to MBus. The isolate (`ISO`) signals ensure that floating signals from power-gated blocks remain at stable defaults. Systems that do not perform power-gating omit these isolation gates and all of the green blocks.

Module	Verilog SLOC	Gates	Flip-Flops	Area in 180 nm
Bus Controller	947	1314	207	27,376 μm^2
Optional				
Sleep Controller	130	25	4	3,150 μm^2
Wire Controller	50	7	0	882 μm^2
Interrupt Controller	58	21	3	2,646 μm^2
Total	1185	1367	214	37,200 μm^2 §
	Other Buses:			
SPI Master†	516	1004	229	37,068 μm^2
I^{2}C ‡	720	396	153	19,813 μm^2
Lee I^{2}C [14]	897	908	278	33,703 μm^2

§ Includes a small amount of additional integration overhead area
† SPI Master from OpenCores [32] synthesized for our 180 nm process
‡ I^{2}C Master from OpenCores [10] synthesized for our 180 nm process

Table 2: Size of MBus Components. Non power-gated designs require only the Bus Controller. The MBus values are from the temperature sensor chip in Figure 12. To support its additional features and lower power, MBus incurs a modest increase in area.

6.1. Protocol Evaluation

Topology. Because MBus is a ring, as the number of nodes increases, so does the propagation delay around the ring. The MBus specification defines a maximum node-to-node delay of 10 ns, which is achieved by all of our designs. Figure 9 explores how node count affects the bus clock and finds that a 14-node MBus system can run at up to 7.1 MHz. I^{2}C clock speed ranges from 100 kHz (Standard) to 5 MHz (Ultra Fast) [23]. Some special-purpose SPI implementations reach speeds as high as 100 MHz, though most low-power microcontrollers have an upper limit of 16 MHz for the I/O clock [5, 34].

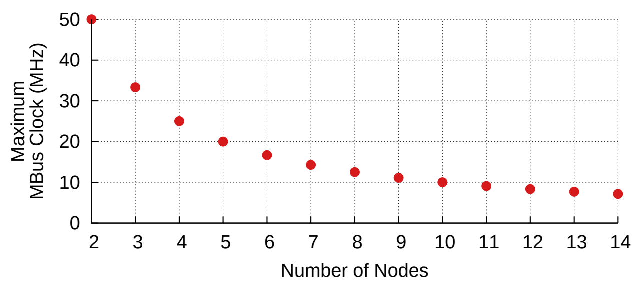


Figure 9: Maximum Frequency. MBus peak clock frequency is inversely proportional to the number of nodes. MBus limits node-to-node propagation delay to 10 ns. For the maximum of 14 short-addressed nodes, MBus could support a 7.1 MHz bus clock.

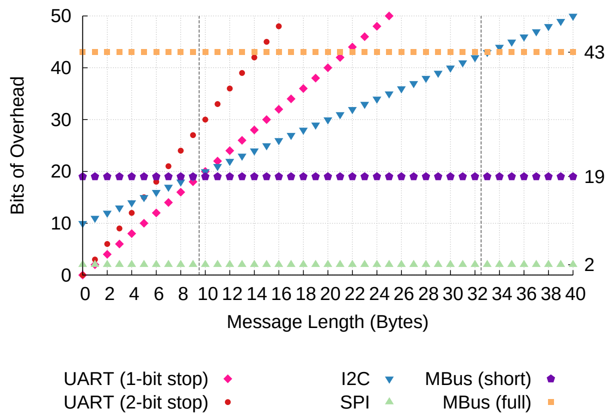


Figure 10: Bus Overhead. MBus message overhead is independent of message length. MBus short-addressed messages become more efficient than 2-mark UART after 7 bytes and more efficient than I^2C and 1-mark UART after 9 bytes. MBus scales efficiently to messages such as 28.8 kB images (Section 6.3.2) or longer.

Overhead. In addition to transmitting data, MBus transactions require arbitration (3 cycles), addressing (8 or 32 cycles), interjection (5 cycles), and control (3 cycles), an overhead of 19 or 43 cycles depending on the addressing scheme. Figure 10 compares MBus overhead to other common buses and finds that MBus's length-independent overhead is more efficient after 9 byte payloads than length-dependent protocols, without incurring significantly greater overhead for shorter messages.

6.2. Power

Simulation. To capture a detailed estimate of the cost of running MBus, we run our implementation through Synopsis PrimeTime. Our model is post-APR and uses standard wire models. We choose a conservative pad model, estimating 2 pF per pad. Our simulation estimates that MBus draws 5.6 pW per chip in idle and consumes 3.5 pJ/bit/chip while transmitting. We estimate the energy of a single MBus message as:

$$E_{message} = [3.5\,\text{pJ} * (\{19 \text{ or } 43\} + 8 * n_{bytes})] * n_{chips}$$

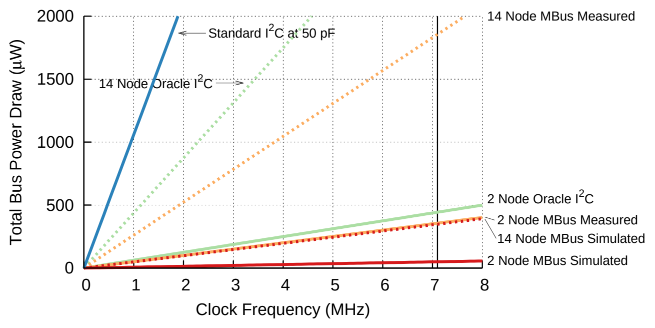
		Energy per bit
Member+Mediator Node	sending	27.5 pJ/bit
Member Node	receiving	22.7 pJ/bit
Member Node	forwarding	17.6 pJ/bit
Average		22.6 pJ/bit

Table 3: Measured MBus Power Draw. Using differential system states, we measure an estimate of the power draw for MBus. Forwarding nodes reduce switching activity by not clocking flops in their receive buffer. The mediator is integrated as a block in our processor and cannot be isolated.

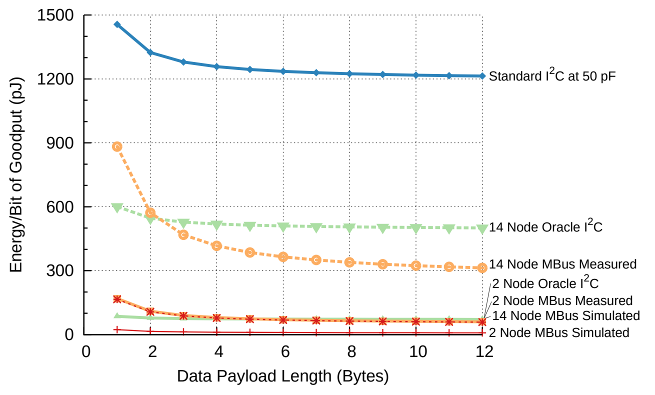
Measurement. We perform empirical power measurements on a debug (non-stacked) version of the temperature sensor shown in Figure 12 and evaluated in Section 6.3.1. Although we cannot directly measure the power draw of MBus in our fabricated chips, we can measure the draw of each chip in the system in different states. The mediator is integrated as a block on our processor chip and cannot be disentangled from the rest of that chip's power draw. To get stable measurements, we place the processor in a continuous loop sending invalid commands to the sensor node, which ignores them. As only the processor can be configured to send continuous messages, we can report only the combined mediator and transmit energy consumption. The results of our measurements are summarized in Table 3. The simulation number is only the energy consumed by MBus directly. We attribute the ~6.5× increase over simulation to overhead such as internal memory buses and other integrated components that could not be isolated.

We have no means to directly or indirectly measure the power draw of MBus when idle. The total idle power draw of the temperature system is 8 nW, three orders of magnitude above the expected static leakage of MBus (5.6 pW), and comparable with the idle system power of the prior state-of-the-art. Thus, we conclude that MBus contributes negligible power to the idle state.

Comparison to I^2C. The biggest inefficiency in I^2C stems from overprovisioning. Since the total bus capacitance is unknown, a power-inefficient, smaller value resistor must be chosen to guarantee timing constraints are met. We imagine an "Oracle I^2C", in which the exact bus capacitance is known and an ideally large resistor is selected. To further improve Oracle I^2C power performance, we allow the rise time to take the entire half clock period (zero setup and hold time) and treat 80% V_{DD} as logical `1`. We model Oracle I^2C using the same simulation parameters as MBus (1.2 V, 2 pF/pad, 0.25 pF/wire). We compare the performance of our MBus simulation, an extrapolation of our measured MBus values, Oracle I^2C simulation, and standard I^2C in Figure 11. Both simulated and measured MBus outperform Oracle I^2C for all but the shortest ($1-2$ byte) messages.



(a) Total Power Draw



(b) Energy of Goodput Bits

Figure 11: Energy Comparisons. In (a) we compare the power draw of various bus configurations as clock frequency and node population increase. We find that both our simulated and measured MBus outperforms the simulated Oracle I^{2}C, which itself outperforms standard I^{2}C. In (b) we examine MBus overhead by computing the energy per bit for each bit of goodput, actual data bits that amortize protocol overhead. Our simulated MBus outperforms the simulated Oracle I^{2}C for all payload lengths. Our measured MBus reveals that MBus efficiency suffers for short (1–2 byte) messages and that systems should attempt to coalesce messages if possible. In both figures, the measured values are based on empirical measurements of our 3-node temperature sensor.

6.3. Microbenchmarks

We next examine two systems, representative of typical embedded workloads, and demonstrate the importance of multi-master capability, efficient handling of large messages, and power-conscious design.

6.3.1. Sense and Send. Figure 12 shows our temperature sensor. This system is an archetypal "sense and send" design. The environment is periodically sampled and the reading is communicated from the sensor. In our system, the processor node periodically requests a temperature reading from the sensor node. In the request (4 bytes), the sensor node can be instructed to send the response (8 bytes) directly to the radio node, which transmits the message. These requests are infrequent (every 15 s) and short in duration, leading to a bus utilization of only 0.0022% at 400 kHz.

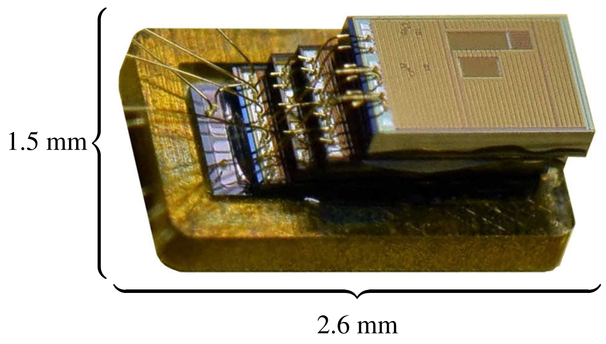
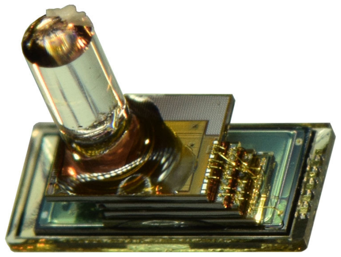
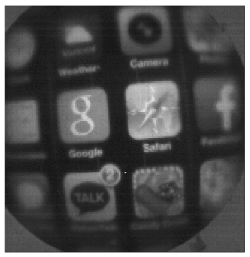


Figure 12: Temperature Sensing System. A system we designed consisting of a 2 µAH battery, a 900 MHz near-field radio, an ARM Cortex M0 processor, and an ultra-low power temperature sensor, interconnected using MBus.



(a) Integrated System



(b) Captured Image

Figure 13: Motion Detection and Imaging System. (a) Our imager made of a 900 MHz near-field radio, a 5 µAH battery, an ARM Cortex M0, and a 160×160 pixel, 9-bit graysacle imager with ultra-low power motion detection all connected using MBus. (b) A full-resolution (28.8 kB) image that was transferred by MBus.

While transmitting the message directly from the sensor to the radio does reduce total bus utilization by 40%, that resource is not contested in this case. Power, however, is always a concern. In this three-chip stack we have one sender, one receiver, and one forwarder; sending an 8 byte message requires

$$(64\,\text{bits} + 19\,\text{bits}) \times (27.45\,\frac{\text{pJ/bit}}{\text{TX}} + 22.71\,\frac{\text{pJ/bit}}{\text{RX}} + 17.55\,\frac{\text{pJ/bit}}{\text{FWD}})$$

$= 5.6\,\text{nJ}$; sending it twice would require 11.2 nJ. Further energy savings come from not powering on the processor. Our processor uses ~20 pJ/cycle and requires ~50 cycles to handle an interrupt and copy an 8 byte message to be sent again, using

$$50\,\text{cycles} \times 20\,\frac{\text{pJ}}{\text{cycle}}$$

$= 1\,\text{nJ}$. To estimate the energy cost of an entire sense and send sequence, we place the system in a continuous loop and measure the average power draw and sample rate. We find that each sense and send event requires about 100 nJ of energy. By supporting any-to-any communication, MBus reduces the energy consumption of each sense and send event by 6.6 nJ (~7%). Using the crude battery capacity of approximation of $2\,\mu\text{Ah} \times 3.8\,\text{V} = 27.4\,\text{mJ}$, for a 15 s sample interval this increases node lifetime by 71 hours, from ~44.5 to ~47.5 days.

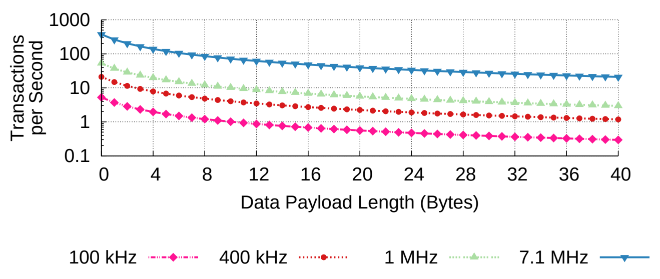


Figure 14: Saturating Transaction Rate. As a shared medium, MBus can only support a finite number of transactions across all member nodes. The peak transaction rate depends on the transaction size and bus clock speed.

6.3.2. Monitor and Alert Our second system, the motion-activated camera seen in Figure 13, exemplifies a typical monitor, filter, and alert system, and it demonstrates the need and efficacy of MBus's power faculties and efficient handling of large messages. During ultra-low power motion detection, the imager power-gates nearly all of its logic to minimize leakage. When motion is detected, the motion detector simply needs to assert one wire for MBus to wake the chip. By decoupling power management, the motion detector may act as a simple, standalone circuit or as a trigger to enter the more power-hungry image capture state whenever motion is detected.

The imager itself is a 160×160 pixel CMOS camera with 9-bit single-channel (grayscale) resolution. A full resolution image is 28.8 kB. Our implemented MBus clock is run-time tunable from 10 kHz to up to 6.67 MHz (default is 400 kHz). Transferred as a single message, a full resolution image could take from 4.2 ms (238 fps) to 2.9 s (0.3 fps) depending on clock speed. Like most CMOS imagers, however, our camera reads pixels out one row at a time. To better cooperate with other possible bus users, the camera sends each row as a separate message, with small delays in-between while the next row is read out. Recall the correlation from Figure 10 between MBus message length and efficiency. By sending 160 180-byte messages instead of one 28.8 kB message, the image transmission incurs an additional 3,021 bits or 1.31% of overhead. By comparison, I^2C would incur 28,810 bits (12.5%) of overhead transmitting the whole image and 30,400 bits of overhead (13.2%) if sent row-by-row. MBus's message-oriented acknowledgment protocol results in a $90-99\%$ reduction in overhead compared to a byte-oriented approach.

6.4. Many-Node Systems

Both of our microbenchmarks are fundamentally one-sensor systems. One possible concern of the MBus design is how well it will scale to a greater number of connected nodes and what the impact on lower priority nodes will be. A large and shared bus, ring, or interconnect topology, is not uncommon in an embedded design. Most microcontrollers have only one or two I^2C and/or SPI interfaces. I^2C also has a fixed priority scheme, based on the target address instead of the physical location of the sender. SPI is more flexible, allowing the central controller to select a priority scheme dynamically, at the cost of requiring a central controller to do so. The more important metric is not node count, rather it is the desired transaction rate – barring protocol overhead, for any bus two nodes sending messages at 1 Hz yields the same utilization as one node sending at 2 Hz. Figure 14 considers possible rates of MBus transactions as a function of message length. For brief periods of burst transactions that exceed the saturation rate, MBus provides both physical and logical mechanisms to enable system designers to federate bus access.

6.5. Interoperability

A key design goal was to facilitate interoperability independent of the technology used to fabricate MBus without requiring any tuning or tweaking. As a series of singled-ended connections—totem-pole FETs driving gates—the MBus design is well-suited to meet this constraint. As evidence of this claim, our systems integrate chips from 65, 130, and 180 nm processes from two different fabs. We also verify operation with debug interfaces from an NI board [22] and a Microsemi IGLOO nano FPGA [20]. Newer MBus chips add built-in level converters for I/O pins, however all of the chips tested in this work operated at 1.2 V. In over 1,000 hours of system testing, we are yet to encounter any MBus-related issues.

6.6. Bitbanging MBus.

To investigate MBus viability on existing microcontrollers without a dedicated MBus interface, we implement MBus in C. Our implementation is general and requires only four GPIO pins (two must have edge-triggered interrupt support). To estimate the overhead of bitbanging MBus, we target an MSP430 [34] using `msp430-gcc-4.6.3` [35] and find that our worst case path is 20 instructions (65 cycles including interrupt entry and exit) to drive an output in response to an edge. With an 8 MHz system clock speed, the MSP430 can support up to a 120 kHz MBus clock. For a comparison, we compile[1] Wikipedia's I^2C bitbang implementation and find it has similar overhead with a longest path of 21 instructions [2].

7. Discussion

We elide many small details and corner cases in this paper that will be covered in the MBus Specification. In this section we discuss some of the potential concerns as well as some ideas for future directions for MBus.

Topological Priority, Fairness, and Progress. Currently, the mediator always has top priority. One could imagine a mutable priority scheme, however, where the node assigned to not forward during arbitration is assigned dynamically, e.g. by a broadcast configuration message. Mutable priority would require adding state to the always-on Wire Controller, however.

[1] All of the stub functions (e.g. `read_SCL()`) were converted into direct memory accesses assuming a single memory operation MMIO interface.

MBus does not guarantee fairness (nor does I^2C). Making arbitration fair a priori is difficult (how should ties be broken?) and in many cases, system designers may prefer prioritization over fairness, e.g. the automotive CAN bus, in which braking has higher priority than windshield wipers. If mutable priority is available, one fair scheme could automatically rotate priority on every message.

MBus expressly does not introduce any form of time-based backoff scheme. MBus is designed to support nodes with no sense of local time and any formal backoff scheme would violate this. To ensure progress, as a matter of policy MBus requires that a node that wins arbitration be permitted to send at least four bytes before being interrupted. Our Bus Controller implementation enforces this policy.

Long Messages and Latency. MBus permits arbitrary length messages. While this is useful for sending long messages efficiently, it also has the effect of locking the bus for a long time, which may harm responsiveness. While the design of MBus lends itself well to resuming an interrupted transmission (both TX and RX nodes know how far through a message they were), it is not possible to indicate to other nodes on the bus which messages are interrupt-friendly. One idea is to leverage one or more functional units as well-known resumable message destinations to indicate to all nodes that this message may be opportunistically interrupted. However, resumable messages comes with other challenges – nodes must have buffer(s) for multiple in-flight transactions and preserve state across transactions, complicating bus controller design.

Runaway Messages. Since MBus messages have unbounded length, a transmitter in a bad state could lock the bus by transmitting forever. To address this, the mediator includes a counter that imposes a maximum message length on the local system. MBus requires a minimum maximum length of 1 kB. This value is set via a broadcast message on the configuration channel, allowing all interested nodes to track it (along with other MBus configuration, such as clock speed).

Increasing Bandwidth. As a ring with a (small) number of gates between `IN` and `OUT` at each node, the upper bound for clock speed is limited. However, the design is amenable to additional `DATA` lines. While this increases I/O cost, each additional `DATA` line doubles the MBus payload throughput. Figure 15 examines the goodput, actual data transmitted, of parallel MBus if data transmission is parallelized but other protocol elements are unchanged. A hybrid ring of traditional and parallel MBus nodes can be imagined, with the `DATA0` line touching every node and the additional data lines forming a smaller ring of only parallel-capable MBus nodes. When transmitting, the bus controller could stripe data across as many `DATA` lines as the target node has available. This parallel MBus design is backward compatible with traditional MBus and can even use the same, unmodified mediator.

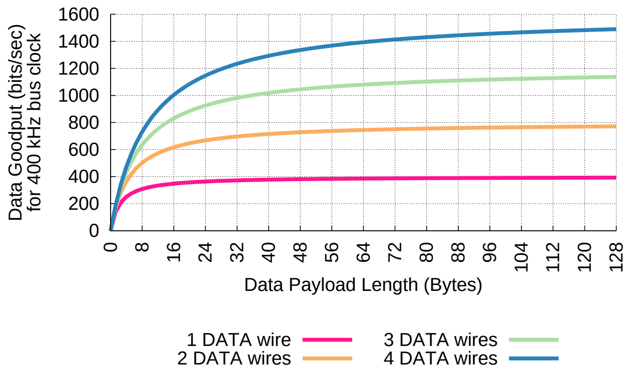


Figure 15: Parallel MBus Goodput. To increase bandwidth without increasing clock speed, parallel `DATA` lines could be used. For very short messages, MBus protocol overhead dominates goodput. As protocol overhead is independent of message length, goodput improves as the message length grows.

8. Conclusion

Despite the advantages of reusable and flexible design that modular components have provided for intermediate computing classes, recent micro-components have not been designed with reusability in mind. We investigate this phenomenon and claim that one cause is the absence of a suitable chip interconnect technology to integrate these components into a complete system—one that respects their ultra-low power constraints, need for configuration flexibility, and limited silicon area. To address this need, we design, implement, and evaluate MBus, a new chip-to-chip interconnect for micro-scale systems. MBus both fills a void for micro-scale systems and bridges the adoption gap with existing technology by supporting seamless interoperation between power-conscious and power-oblivious devices. Implemented on a dozen micro-scale chips, MBus demonstrates a viable chip interconnect design point for next generation nano-power systems.

9. Acknowledgments

The authors would like to thank Bradford Campbell and Mark Horowitz for their ideas and feedback, including the elegant rotating priority scheme in the discussion. This paper is much stronger for their assistance. This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program sponsored by MARCO and DARPA. This research was conducted with Government support under and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a. This material is based upon work partially supported by the National Science Foundation under grants CNS-0964120, CNS-1111541, and CNS-1350967, and generous gifts from Intel, Qualcomm, and Texas Instruments.

References

[1] "System management bus BIOS interface specification," http://smbus.org/specs/smbb10.pdf.

[2] "Wikipedia: Example of bit-banging the I^{2}C master protocol," http://en.wikipedia.org/wiki/I2c#Example_of_bit-banging_the_I.C2.B2C_Master_protocol, Nov. 2013.

[3] Actel, "SmartFusion: Customizable System-on-Chip (cSoC)," http://www.actel.com/documents/SmartFusion_DS.pdf.

[4] J. Brown, K.-K. Huang, E. Ansari, R. Rogel, Y. Lee, and D. Wentzloff, "An ultra-low-power 9.8 GHz crystal-less UWB transceiver with digital baseband integrated in 0.18 BiCMOS," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, 2013, pp. 442–443.

[5] Byte Paradigm, "SPI Storm datasheet," http://www.byteparadigm.com/files/documents/ds_SPIStorm.pdf, dec 2011.

[6] P. Chu, N. Lo, E. Berg, and K. Pister, "Optical communication using micro corner cube reflectors," in *Micro Electro Mechanical Systems, 1997. MEMS '97, Proceedings, IEEE., Tenth Annual International Workshop on*, 1997, pp. 350–355.

[7] M. Crepaldi, C. Li, K. Dronson, J. Fernandes, and P. Kinget, "An ultra-low-power interference-robust IR-UWB transceiver chipset using self-synchronizing OOK modulation," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, 2010, pp. 226–227.

[8] V. Ekanayake, C. Kelly, and R. Manohar, "An ultra low-power processor for sensor networks," in *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XI. New York, NY, USA: ACM, 2004, pp. 27–36.

[9] S. Hanson and D. Sylvester, "A 0.45–0.7 V sub-microwatt CMOS image sensor for ultra-low power applications," in *VLSI Circuits, 2009 Symposium on*, June 2009, pp. 176–177.

[10] R. Herveille, "I^{2}C-Master core specification," svn://opencores.org/ocsvn/i2c@r76, OpenCores.

[11] U. Kiencke, S. Dais, and M. Litschel, "Automotive serial controller area network," *SAE Technical Paper 860391*, 1986.

[12] R. Kumar and G. Hinton, "A family of 45 nm IA processors," in *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, 2009, pp. 58–59.

[13] Y.-S. Kuo, P. Pannuto, G. Kim, Z. Foo, I. Lee, B. Kempke, P. Dutta, D. Blaauw, and Y. Lee, "MBus: A 17.5 pJ/bit/chip portable interconnect bus for millimeter-scale sensor systems with 8 nW standby power," in *Custom Integrated Circuits Conference (CICC), 2014 IEEE Proceedings of the*, Sept 2014, pp. 1–4.

[14] Y. Lee, S. Bang, I. Lee, Y. Kim, G. Kim, M. H. Ghaed, P. Pannuto, P. Dutta, D. Sylvester, and D. Blaauw, "A modular 1 mm^3 die-stacked sensing platform with low power I^{2}C inter-die communication and multi-modal energy harvesting," in *IEEE Journal of Solid-State Circuits*, vol. 48, 2013.

[15] Y. Lee, B. Giridhar, Z. Foo, D. Sylvester, and D. Blaauw, "A sub-nW multi-stage temperature compensated timer for ultra-low-power sensor nodes," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 10, pp. 2511–2521, 2013.

[16] Y.-S. Lin, D. Sylvester, and D. Blaauw, "A sub-pW timer using gate leakage for ultra low-power sub-Hz monitoring systems," in *Custom Integrated Circuits Conference, 2007. CICC '07. IEEE*, 2007, pp. 397–400.

[17] Y.-S. Lin, D. Sylvester, and D. Blaauw, "An ultra low power 1 V, 220 nW temperature sensor for passive wireless applications," in *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE*, 2008, pp. 507–510.

[18] M. Mansuri, J. Jaussi, J. Kennedy, T. Hsueh, S. Shekhar, G. Balamurugan, F. O'Mahony, C. Roberts, R. Mooney, and B. Casper, "A scalable 0.128-to-1 Tb/s 0.8-to-2.6 pJ/b 64-lane parallel i/o in 32 nm CMOS," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, Feb 2013, pp. 402–403.

[19] Microsemi, "IGLOO nano FPGAs," http://www.microsemi.com/products/fpga-soc/fpga/igloo-nano.

[20] Microsemi, "IGLOO nano low power flash FPGAs," http://www.actel.com/documents/IGLOO_nano_DS.pdf.

[21] T. Nakagawa, M. Miyazaki, G. Ono, R. Fujiwara, T. Norimatsu, T. Terada, A. Maeki, Y. Ogata, S. Kobayashi, N. Koshizuka, and K. Sakamura, "1-cc computer using UWB-IR for wireless sensor network," in *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*, 2008, pp. 392–397.

[22] National Instruments, "NI PCIe-6535 10 MHz digital I/O for PCI Express," http://sine.ni.com/nips/cds/view/p/lang/en/nid/205628.

[23] NXP, "I^{2}C-bus specification Rev. 6," http://www.nxp.com/documents/user_manual/UM10204.pdf, Apr 2014.

[24] P. Pannuto, Y. Lee, B. Kempke, D. Sylvester, D. Blaauw, and P. Dutta, "Demo: Ultra-constrained sensor platform interfacing," in *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, ser. IPSN '12. New York, NY, USA: ACM, 2012, pp. 147–148. Available: http://doi.acm.org/10.1145/2185677.2185721

[25] J. Poulton, W. Dally, X. Chen, J. Eyles, T. Greer, S. Tell, J. Wilson, and C. Gray, "A 0.54 pJ/b 20 Gb/s ground-referenced single-ended short-reach serial link in 28 nm CMOS for advanced packaging applications," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 12, pp. 3206–3218, Dec 2013.

[26] A. Ricci, M. Grisanti, I. De Munari, and P. Ciampolini, "Improved pervasive sensing with RFID: An ultra-low power baseband processor for UHF tags," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 12, pp. 1719–1729, 2009.

[27] V. Sathe, S. Arekapudi, A. Ishii, C. Ouyang, M. Papaefthymiou, and S. Naffziger, "Resonant-clock design for a power-efficient, high-volume x86-64 microprocessor," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 1, pp. 140–149, Jan 2013.

[28] M. Scott, B. Boser, and K. Pister, "An ultralow-energy ADC for Smart Dust," *Solid-State Circuits, IEEE Journal of*, vol. 38, no. 7, pp. 1123–1129, 2003.

[29] M. Seok, S. Hanson, Y.-S. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw, "The Phoenix Processor: A 30 pW platform for sensor applications," in *VLSI Circuits, 2008 IEEE Symposium on*, 2008, pp. 188–189.

[30] M. Seok, S. Hanson, Y.-S. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw, "Phoenix: An ultra-low power processor for cubic millimeter sensor systems," *ACM/IEEE Design Automation Conference (DAC)*, 2009.

[31] Y. Shapira, "Tel Aviv University Review," http://english.tau.ac.il/sites/default/files/media_server/TAU%20Review%202008-09.pdf, Smart Dew.

[32] S. Srot, "SPI Master core specification," svn://opencores.org/ocsvn/spi@r29, OpenCores.

[33] N. Sturcken, E. O'Sullivan, N. Wang, P. Herget, B. Webb, L. Romankiw, M. Petracca, R. Davies, R. Fontana, G. Decad, I. Kymissis, A. Peterchev, L. Carloni, W. Gallagher, and K. Shepard, "A 2.5D integrated voltage regulator using coupled-magnetic-core inductors on silicon interposer," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 1, pp. 244–254, Jan 2013.

[34] Texas Instruments, "MSP430F161x Mixed Signal Microcontroller," http://www.ti.com/product/msp430f1611.

[35] Texas Intruments and Red Hat, "GCC – Open source compiler for MSP430 microcontrollers," http://www.ti.com/tool/msp430-gcc-opensource.

[36] B. Warneke, B. Atwood, and K. Pister, "Smart dust mote forerunners," in *Micro Electro Mechanical Systems, 2001. MEMS 2001. The 14th IEEE International Conference on*, 2001, pp. 357–360.

[37] B. Warneke and K. 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*, 2004, pp. 316–317 Vol.1.

[38] S. G. Younis, "Asymptotically zero energy computing using split-level charge recovery logic," Ph.D. dissertation, Massachusetts Institute of Technology, Jun 1994, http://hdl.handle.net/1721.1/7058.