MBus: A System Integration Bus for the Modular Microscale Computing Class

Pat Pannuto
University of Michigan

Yoonmyung Lee
Sungkyunkwan University

Ye-Sheng Kuo
ZhiYoong Foo
Benjamin Kempke
Gyouho Kim
Ronald G. Dreslinski
David Blaauw
Prabal Dutta
University of Michigan

MBus is a new interchip interconnect made of two "shoot-through" rings that resolves fundamental size and power issues that prevent the design of composable microscale systems. MBus introduces power-oblivious communication, which guarantees message reception regardless of the recipient's power state. This disentangles power management from communication, greatly simplifying the creation of viable, modular, and heterogeneous systems that operate on the order of nanowatts.

Bell's law follows the trend from mainframes to desktops to mobile phones and predicts the emergence of a new class of computing roughly every decade. Today, we see the emergence of the wearable class: centimeter-scale, wireless, battery-powered devices. We look ahead to the next generation, the microscale computing class, and make the critical observation that even if we scale all the components, the processor, the radio, sensors, and so on, we still will not be able to produce viable microscale systems.

The problem is rooted in the interconnect technologies used to synthesize systems. Microscale systems such as the pressure sensor in Figure 1 by their definition introduce unprecedented constraints on size, which in turn introduces unprecedented constraints on available energy. Since their invention around the early 1980s, the Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I^2C) buses have served as the interconnects for the vast majority of embedded designs. In the creation of modular, microscale systems, we discover that the area demands of SPI and the energy demands of I^2C render them fundamentally unusable for composing the next class of computing.

To address these limitations, we introduce *MBus,* a new interchip interconnect designed for the composition of current and future microscale and resource-constrained systems.[1] MBus connects components with a pair of "shoot-through" rings, `CLK` and `DATA`, and delivers a superset of the features from I^2C and SPI but at lower power, with a fixed area and pin count, using fully synthesizable logic, and minimal protocol overhead. We designed MBus from the ground up, leveraging a principled approach that begins by identifying the key properties of interconnect technologies from extant and newly emerging embedded designs.

Because existing technologies have worked for so long, little to no work outlines the design space and requirements for the inter-chip networks of embedded devices. Our first major contribution was to fill this void and present an analysis of the requirements and desirable features we identified from nearly a decade of designing microscale systems. From this discussion emerged a new constraint, unique to the new but growing class of hyper-power-conscious systems of a "power-aware" interconnect.

With power awareness, MBus takes a challenging circuits problem and a challenging systems problem and creates an architectural solution. To preserve precious energy, microscale systems completely shut off unused components. This is not the traditional dark silicon, which still leaks static power while unclocked. This is "pitch-black" silicon that draws no power but also preserves no state. In circuit design, cold-booting pitch-black, clockless silicon is a challenging problem. The MBus protocol adds explicit support for waking powered-off nodes, enabling circuit designers to leave fewer than 50 gates powered for maximal energy savings. When designing systems, tracking every component's power state can become a challenging distributed state problem. Because MBus controls each component's power state, it can provide the appearance that components are always on simply by waking them before delivering messages. This lets MBus member nodes be power oblivious, which greatly simplifies the system design and enables traditional components that have no power awareness to seamlessly benefit from the power-saving capabilities of power-conscious elements.

We have implemented MBus front ends on more than a dozen distinct chips and used these components to compose half a dozen different systems. From empirical measurements, MBus requires only 22.6 pJ/bit/chip on average, and its efficient transmission-level acknowledgements provide a 90 to 99 percent reduction in overhead for long messages.

Shortcomings of Existing Interconnects

The SPI bus is conceptually very simple. Direct, single-ended connections run between a master and peripheral devices. For efficiency, the clock and data wires (SCLK, MOSI, MISO) are shared among all peripherals, with only one distinct chip select line running to each slave device. As we push toward smaller and more I/O-constrained devices, however, it is this multitude of chip selects that becomes the issue.

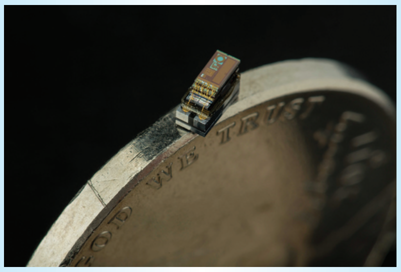
Figure 1. Microscale pressure sensing system on the edge of a US nickel. Microscale systems, and the techniques such as 3D-stacking employed to realize their size, introduce volume and surface area constraints that lead to extremely limited energy storage, energy harvesting, and I/O capabilities.

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 microscale systems. Furthermore, the master/slave architecture actually makes I/O twice as problematic. A subtle, yet critical, implication of a single-master design is that all communication is master-initiated. For a sensor to signal the microcontroller (that is, an interrupt), it requires a second dedicated wire, a resource that is unavailable to microscale systems.

Another drawback of the master/slave architecture is that all communication between slave devices must go through the master node. This more than doubles the communication cost for slave-to-slave transmissions: every message is sent twice, plus the extra energy cost of running the central controller. Owing primarily to I/O constraints, SPI and its derivatives are fundamentally incompatible with size-constrained microsystems.

The other primary embedded interconnect technology, the I^{2}C bus, requires only two wires, clock and data (SCL and SDA), independent of the number of connected devices. Each wire is an open-collector circuit, connected to each device. This 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. This approach has the advantages of decentralized arbitration and multitiered 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 microscale systems; fast mode I^2C has a 400 kHz clock and must reach 80 percent 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, which are outside the scope of our design. MBus finds its energy gains by eliminating the 151 pJ/bit lost to the pull-up resistor.

In an earlier generation, we attempted to modify I^2C, replacing the energy-hungry pull-up resistors with a low-energy bus-keeper circuit that preserves the last value, similar to I^2C's ultrafast mode.[2] Although this improved energy, pushing it down to 88 pJ/bit (four times that of MBus), the design required manual, process-specific tuning for each fabricated chip—not an appealing property for a general-purpose bus. Furthermore, although the design attempted to preserve commercial interoperability, in practice, actual interoperation required a field-programmable gate array to translate between actual I^2C and the "I^2C-like" bus. MBus eschews the "partial compatibility" that I^2C-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.

Interconnect Requirements

Having found show-stopping limitations with existing interconnect technology, we decided to turn the problem on its head. Drawing from our experience as embedded system and microscale system developers, we drew up a list of key properties that we identified for the system interconnect of ultraconstrained systems (see Table 1). Although these properties will eventually guide our development of MBus, they stand independent of the MBus design as an architectural summary of the demands for an interconnect for microscale systems.

Synthesizable

To facilitate widespread adoption, we require a process-agnostic solution, a block of "pure" HDL with no process-specific custom macros. Our previous I^2C variant required process-specific tuning of custom ratioed logic, adding cost, complexity, and risk to every chip.[2]

Low Wire Count

With submillimeter-scale systems, the area cost required to place bonding pads (35 to 65 μm wide) on one edge or around the perimeter of a chip limits the system's ability to scale. Although advancements such as through-silicon vias help, many popular processes do not support them (such as IBM130 or TSMC65).

Address Space

A bus must provide a means of addressing each element, for example, with hardware support (such as SPI chip selects) or explicit addresses (such as I^2C). If addresses are used, they must both support a large number of possible devices (a naive search of DigiKey returns over 2^{15} I^2C-capable devices) and minimize overhead on each transmission.

Low Standby Power

Resource-constrained systems spend most of their time in standby, so standby inefficiencies

Table 1. Population-independent area, ultra-low-power operation, synthesizability, an area-free global namespace, and interrupt support are fundamental requirements for a microscale interconnect.

Key properties Critical	Inter-Integrated Circuit (I^2C)	Serial Peripheral Interface (SPI)	Universal Asynchronous Receiver Transmitter (UART)	Lee-I^2C^2	MBus
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
Globally unique addresses	128	—	—	128	224
Multimaster (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 acknowledgments	Yes	No	No	Yes	Yes
Bits overhead (*n* byte message)	$10 + n$	2	$(2-3) \times n$	$10 + n$	19 or 43

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

Microscale 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 less than 40 μW (and idle power of 20 nW). For reference, recall that the idealized I^2C clock alone uses 69.6 μW. Although any absolute number will be system dependent, to allow an 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 a message's length. 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. These faults include spontaneous bit flips or other glitches, but cannot necessarily account for permanent failures such as faulty hardware.

Interrupts

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 multimaster design.

Efficient Acknowledgments

Many applications require reliable message transport. This feature can 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. Ultraconstrained systems need to frequently cold boot and shut down subcircuits without affecting active areas. To power on a power-gated circuit reliably and without introducing glitches can be a challenging circuit-design problem, but it is made much simpler with a reliable clock source.

Aggressively low-power designs have no clock sources in their lowest-power state, however, so every design requires a custom wakeup circuit to generate these edges, adding cost and complexity. For systems

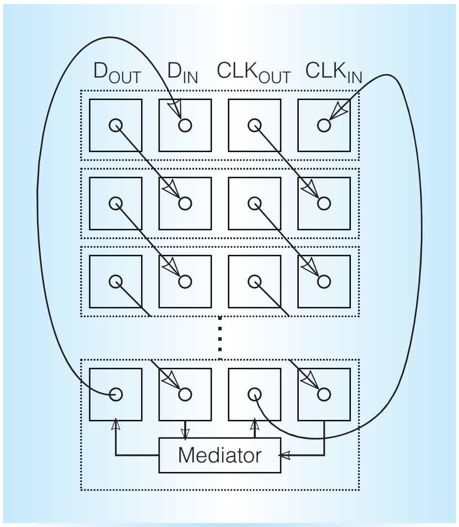

Figure 2. MBus physical topology. An MBus system consists of a mediator node and one or more member nodes, connected in two "shoot-through" rings, CLK and DATA.

developers, managing the system's power state as various subcomponents try to quickly power one another on and off becomes a challenging distributed state problem.

By adding power awareness to the interconnect fabric, we can eliminate difficult challenges for both systems engineers and circuit designers. The system interconnect should provide a clean abstraction for sending messages—that is, one that ensures receipt independent of the target device type or immediate power state. Because the interconnect is responsible for alerting a component that it needs to boot in the first place, it should also provide assistance to the booting device's wakeup circuitry.

Interoperability

Not all systems are severely resource or energy constrained, yet they might 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.

MBus Design

Working from the requirements outlined for the interconnect of microscale systems, we developed a new interconnect system: MBus.

Topology

To meet the wire count requirement, the bus topology must be independent of the number of nodes—that is, adding another node cannot require adding a new wire. Because 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 2. To minimize active power, MBus clocks all bus logic off the bus clock itself, obviating 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.

Clock Generation

MBus introduces one special node, the mediator, which is responsible for generating the MBus clock and resolving arbitration. Every MBus system must have exactly one mediator, either attached to a core device (such as a microcontroller device) or as a stand-alone component (similar to the pull-up resistors in I^2C). For ultra-low-power designs, MBus power gates all but the forwarding drivers (the wire controller) and a minimalist wakeup front end (the 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.

Arbitration

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 DATA_{OUT} low. This propagates around the DATA ring until it reaches the mediator, which does not forward DATA during arbitration. The falling edge on DATA_{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 DATA_{IN} line. If DATA_{IN} is high,

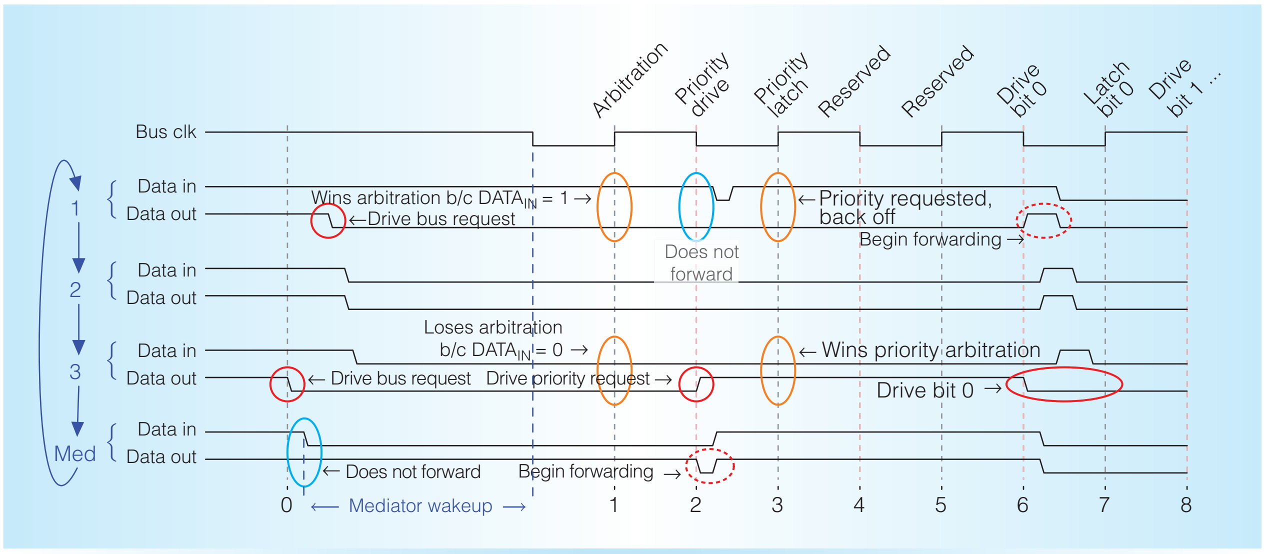

Figure 3. MBus arbitration. To begin a transaction, one or more nodes pull down on $\mathtt{DATA}_{\mathtt{OUT}}$. Here, we show node 1 and node 3 requesting the bus at nearly the same time (node 1 shortly after node 3). Node 1 initially wins arbitration, but node 3 uses the priority arbitration cycle 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.

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 $\mathtt{DATA}_{\mathtt{OUT}}$ high to issue a priority request. Figure 3 shows a waveform of arbitration and priority arbitration.

Hierarchical Wakeup

For maximum power efficiency, MBus on a power-gated node leaves only a minimalist, highly optimized front end 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 to drive the wakeup circuitry of the 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 its address matches. This ensures that a message powers on only the destination node. This design lets any node 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.

Addresses

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), whereas FU-IDs are used to address chip subcomponents. FU-IDs are 4 bits, allowing for up to 16 subcomponents behind each physical MBus front end. 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, which lets nodes listen to only the broadcast messages they support or are interested in.

Prefix Assignment

To retain the efficiency afforded by short addresses while allowing for a diverse ecosystem of unique components, MBus uses runtime 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 side effect 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 can assign themselves a static short prefix, akin to I^2C addressing, so that, if there are no conflicts, enumeration can be skipped.

Every chip design is assigned a unique, 20-bit full prefix. Full prefixes let nodes 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 can be addressed using either short or full addresses interchangeably. It is sometimes advantageous for a system to have two copies of the same chip (for example, memory), which requires short prefixes and enumeration to disambiguate.

Sending Data

MBus transmitters drive data on the falling edge of `CLK`, and receivers latch data on the rising edge of `CLK`. Although standard flops can be clocked only on one edge (rising or falling), only the internal data FIFO needs to be clocked on the falling edge, so this does not violate the synthesizability requirement. This allows for identical setup and hold margins, easing interoperability when driving unknown loads (for more details, see our CICC paper[3]).

At the end of a message, the receiver acknowledges or rejects the entire message. In MBus, any node can terminate any message at any time, even a forwarder. The transmitter might end the message when it is finished, or the receiver might interject midmessage to indicate an error, such as a buffer overrun. Thus, by not interjecting, a receiver implicitly acknowledges every byte. This is less powerful than I^2C acknowledgments, 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.

Ending Messages

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 lets us 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 because the transmitter finished sending data or to express some type of error.

The MBus interjection request mechanism, holding `CLK` high, results in nodes observing a varying number of clock edges depending where they are in the ring. 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 can generate an interjection at any time. This allows for unambiguous signaling of control functions that can resynchronize a bus without requiring out-of-band signals like chip selects or a reset line. This further lets a node with a latency-sensitive message interrupt an active transaction, enabling responsiveness across a diverse array of workloads not possible with current buses.

Evaluation

In our ISCA paper,[1] we evaluated the performance of the MBus protocol in theory and practice and empirically validated the ultra-low-power claims. MBus supports a peak throughput of 7.1 Mbits/second for common designs. Its implicit acknowledgements make it more efficient than I^2C and the Universal Asynchronous Receiver Transmitter (UART) for messages at least 9 bytes long and only slightly (8 bits) less efficient than I^2C for 1-byte messages.

We taped out four chips (processor, radio, temperature sensor, and image sensor) that we composed with MBus into two distinct systems: temperature sensing and imaging (see Figures 4 and 5, respectively). From power traces of the temperature sensor, we found that MBus requires only 22.6 pJ/bit/chip, a four-times reduction over our previous I^2C-like bus and two orders of magnitude better than standard I^2C.

We further evaluated each of the composed systems, showing how MBus's multimaster faculties alone achieve a 7 percent improvement in system lifetime for the temperature sensor. Evaluating the image sensor showcases how well MBus scales to handle (relatively) large volumes of data, such as a 28.8 Kbit image. At that size, MBus can transmit up to 238 frames per second (far more than the imager can produce). The message-level acknowledgements are a key to enabling this image bandwidth, generating a 13 percent reduction in the number of bits sent to transfer images.

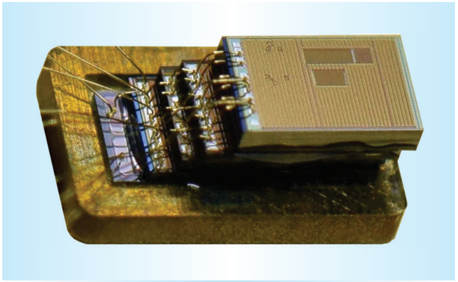
Figure 4. Temperature sensing system. Our evaluated temperature sensing system consisting of a 2 μAH battery, a 900-MHz near-field radio, an ARM Cortex M0 processor, and an ultra-low-power temperature sensor, interconnected using MBus.

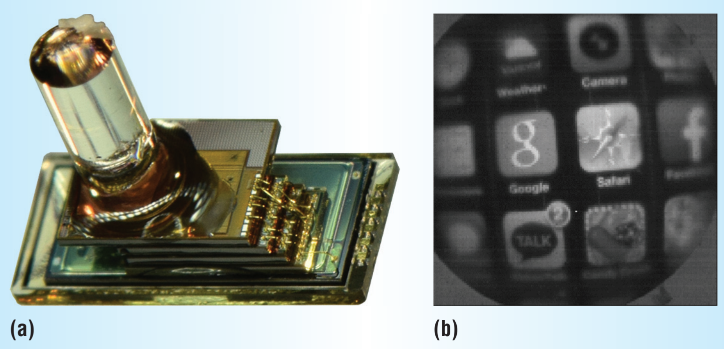
Figure 5. Motion detection and imaging system. (a) Our imager made of a 900 MHz near-field radio, a 5 μAH battery, an ARM Cortex M0, and a 160-$\times$(160-pixel, 9-bit graysacle imager with ultra-low-power motion detection all connected using MBus. (b) A full-resolution (28.8 Kbytes) image that was transferred by MBus.

MBus Limitations, Open Questions, and Future Directions

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 might prefer prioritization over fairness (for example, in the automotive CAN bus, braking has higher priority than windshield wipers). If mutable priority is available, one fair scheme could automatically rotate priority on every message.

Using interjections to end messages permits nodes to send messages of arbitrary length. While this is useful for sending long messages efficiently, it also has the effect of locking the bus for a long time, which could harm responsiveness. Although the MBus design 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 can be opportunistically interrupted. However, resumable messages come with other challenges—nodes must have buffers for multiple in-flight transactions and preserve state across transactions, which complicates bus controller design.

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. Although this increases I/O cost, each additional `DATA` line doubles the MBus payload throughput. 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.

Impact

History has shown time and again that providing a diverse array of powerful building blocks will enable the masses to synthesize unique and creative things undreamed of by the original creators. Today's explosion of intelligent, networked devices (the Internet of Things) is driven, at least in part, by the accessibility of modular components, which let individuals pull together a custom-tailored set of microcontrollers, radios, sensors, and other parts optimized for new applications.

In contrast to these systems, previous forays into microscale design have lacked this modularity. From our own experience, the Michigan Integrated Circuits Lab—the research team that produced MBus and today's Michigan Micro Motes—in 2008 produced the Phoenix processor, then the smallest, lowest-power computing system; it was essentially a temperature sensor. Upon request from collaborators in medicine, the next project was to build a pressure sensor of similar size and energy budget. As a consequence of the tight, monolithic design, it took nearly three years to change a temperature sensing system to a pressure sensing system. MBus introduces critical modularity to the microscale design space. Today, we can develop, optimize, and debug a processor independent of the radio, temperature, or pressure sensor. With MBus, we can pivot the temperature sensing stack presented in our ISCA paper[1] to a pressure sensing stack in only three months, reusing 80 percent of the components.

MBus's key contribution is the recognition that modularity is critical for system design and that no existing technology can enable modularity for microscale systems. MBus (or perhaps a different, new interconnect that respects the requirements of microscale systems) is a necessary and critical enabler for the development of the next class of computing.

Although we have derived great benefit from MBus, it is not a success if it is a technology used exclusively by a team of researchers at one university. For this reason, we released the MBus specification and a reference Verilog implementation for free to the public domain (http://mbus.io).

Today, it is difficult if not impossible to purchase a microcontroller without support for SPI, I^{2}C, or UART. In 20 years, we envision MBus joining that list. Today, we continue to develop new chips that add new capabilities to microscale systems. Leveraging the modularity and reusability afforded by MBus, we can synthesize completely new systems in a fraction of the time previously required. As technology progresses and interest in the design and development of millimeter-scale systems grows beyond the research realm, we aim to establish MBus as the solution for next-generation systems integration.

Currently, as key core components of the Michigan Micro Mote platform stabilize, we are beginning the outreach effort. By providing core components with MBus front ends such as the processor, radio, and power management to system developers, we aim to enable others to springboard rapidly into microscale system design. By releasing MBus itself as a free and open standard with a freely available implementation, we aim to encourage its use and adoption by the emerging microscale systems design community, leading to eventual integration with the broader embedded systems community as microscale systems become mainstream.

Last spring, the Michigan Micro Mote was inducted into the Computer History Museum as the world's smallest computer. It is our sincerest hope that the record does not stand for long, and we expect that MBus will play a key role in helping to continue to break records in the years to come. MICRO

Acknowledgments

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 the DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, and 32 CFR 168a. This material is based on 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. P. Pannuto et al., "MBus: An Ultra-Low Power Interconnect Bus for Next Generation Nanopower Systems," *Proc. 42nd Int'l Symp. Computer Architecture*, 2015, pp. 629–641.
2. Y. Lee et al., "A Modular 1 mm^3 Die-Stacked Sensing Platform with Low Power I^{2}C Inter-Die Communication and Multi-Modal Energy Harvesting," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, 2013, pp. 229–243.
3. Y.-S. Kuo et al., "MBus: A 17.5 pJ/bit/chip Portable Interconnect Bus for Millimeter-Scale Sensor Systems with 8 nW Standby Power," *IEEE Proc. Custom Integrated Circuits Conf.*, 2014; doi:10.1109/CICC.2014.6946046.

Pat Pannuto is a PhD candidate in the Computer Science and Engineering Department at the University of Michigan. His research interests include systems design for microscale systems and technologies for high-fidelity indoor localization. Pannuto received a BS in computer engineering from the University of Michigan. He also received the National Defense Science & Engineering Graduate Fellowship, the National Science Foundation Graduate Research Fellowship Program Fellowship, and the Qualcomm Innovation Fellowship. Contact him at ppannuto@umich.edu.

Yoonmyung Lee is an assistant professor in the Department of Semiconductor Systems Engineering at Sungkyunkwan University. His research interests include energy-efficient integrated circuits design for low-power, high-performance VLSI systems and millimeter-scale wireless sensor systems. Lee received a PhD in electrical engineering from the University of Michigan. He also received a Samsung Scholarship and Intel PhD Fellowship. Contact him at yoonmyung@skku.edu.

Ye-Sheng Kuo is a research fellow in the Computer Science and Engineering Department at the University of Michigan. His research interests include embedded systems, sensor networks, and visible light communication. Kuo received a PhD in electrical engineering from the University of Michigan. Contact him at samkuo@umich.edu.

Zhi Yoong Foo is the Chief Executive Officer of CubeWorks, a startup spun out of the University of Michigan. His research interests include low-cost and low-power VLSI circuit systems integration. Foo received a PhD in electrical engineering from the University of Michigan. Contact him at zhiyoong@umich.edu.

Benjamin Kempke is a PhD candidate in the Electrical Engineering and Computer Science and Engineering Departments at the University of Michigan. His research interests include the design of low-power and high-accuracy indoor RF localization technologies. Kempke received an MSE in computer science and engineering from the University of Michigan. Contact him at bpkempke@umich.edu.

Gyouho Kim is a research fellow in the Electrical Engineering Department at the University of Michigan. His research interests include ultra-low power VLSI design for energy-constrained systems and novel sensing platforms. Kim received a PhD in electrical engineering from the University of Michigan. Contact him at gyouho@umich.edu.

Ronald G. Dreslinski is an assistant professor in the Computer Science and Engineering Department at the University of Michigan.

His research interests include near-threshold computing (NTC), architectural simulator development, and high-radix on-chip interconnects. Dreslinski received a PhD in computer science and engineering from the University of Michigan. He is the winner of the ISSCC 2011 student design contest and is the recipient of the Young Computer Architect Award from the IEEE Computer Society's Technical Committee on Computer Architecture. Contact him at rdreslin@umich.edu.

David Blaauw is a professor in the Electrical and Computer Engineering Department at the University of Michigan. His research interests include VLSI design, ultra-low power, and high performance design. Blaauw received a PhD in computer science from the University of Illinois, Urbana-Champaign. He is an IEEE Fellow. Contact him at blaauw@umich.edu.

Prabal Dutta is an associate professor in the Electrical Engineering and Computer Science Department at the University of Michigan. His research interests include design, deployment, and scaling of wireless, embedded, networked, and sensory systems for applications in health, energy, and the environment. Dutta received a PhD in computer science from the University of California, Berkeley. He is the recipient of an Alfred P. Sloan Research Fellowship, an NSF CAREER Award, a Popular Science Brilliant Ten Award, and an Intel Early Career Award. Contact him at prabal@eecs.umich.edu.