Poster Abstract: Predictable Wireless Embedded Platforms

Felix Sutton, Reto Da Forno, Marco Zimmerling, Roman Lim, Tonio Gsell,
Federico Ferrari, Jan Beutel, and Lothar Thiele
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
{firstname.lastname}@tik.ee.ethz.ch

ABSTRACT

Resource interference is a fundamental barrier to realizing predictable wireless embedded systems. We address this problem by *(i)* partitioning application and communication tasks onto dedicated platforms, and *(ii)* designing a platform interconnect to facilitate asynchronous message exchange with predictable run-time behavior. We motivate the need for this platform interconnect, termed BOLT, and describe a prototype implementation. Evaluation results indicate that the developed platform interconnect exhibits tightly bounded run-time execution with low jitter, and a negligible resource overhead with respect to state-of-the-art application and communication platforms.

Categories and Subject Descriptors

C.0 [**Computer Systems Organization**]: System Architectures

Keywords

Asynchronous message passing, cyber-physical systems.

1. MOTIVATION

The prevalence of Cyber-Physical Systems (CPSs) is encouraging the adoption of wireless embedded systems to control and monitor physical processes within the world around us. The continuum of time is deeply ingrained within these physical processes, and this fact alone, poses a significant challenge in realizing real-world CPS deployments. In particular, to successfully control or monitor such physical processes, software-driven embedded systems must complete a multitude of tasks (*i.e.*, sensing, actuation and communication) within strict timing constraints [2].

Guaranteeing the time correctness of software with today's wireless embedded platforms is extremely difficult to achieve. The common approach is to apply real-time scheduling to enforce a statically-defined or run-time adaptive partitioning of CPU execution time to competing software tasks.

IPSN '15, Apr 14-16, 2015, Seattle, WA, USA
ACM 978-1-4503-3475-4/15/04.
http://dx.doi.org/10.1145/2737095.2737156.

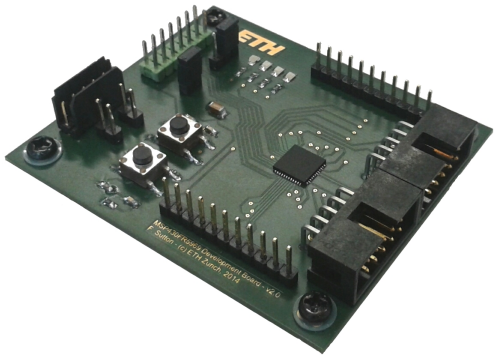


Figure 1: The platform interconnect, BOLT, for enabling predictable wireless embedded platforms.

However, such scheduling schemes are infeasible due to the non-determinism ever present with the physical processes being controlled or monitored. Application software tasks (*e.g.*, reading a sensor, processing data, controlling an actuator) and communication software tasks (*e.g.*, sending and receiving packets) must compete for single resources, *i.e.*, clock cycles, memory and peripheral bus, which cause resource interference. This interference is exacerbated by the fact that the underlying physical process is by nature, non-deterministic. Hence, making it impractical to determine tight worst-case software execution times that take into account all possible environmental interactions and combinations thereof. The ability to provide predictable timing correctness of wireless embedded systems is a key barrier to the successful deployment of large-scale CPS, and one that must be addressed with urgency.

We present a solution to this problem at the platform architecture-level, by *(i)* separating application-specific tasks and communication-specific tasks onto dedicated platforms, thereby removing the resource interference associated with today's single-processor platforms, and *(ii)* introducing a platform interconnect, termed BOLT, for facilitating information exchange between application and communication platforms with guaranteed run-time behavior. As described in the next section, BOLT exhibits favorable properties such as *predictable* timing behavior, as well as *consistent* and *persistent* information storage. We argue that these properties are necessary for the realization of predictable wireless embedded systems.

2. DESIGN & IMPLEMENTATION

The platform interconnect provides a means for application and communication platforms to bi-directionally share

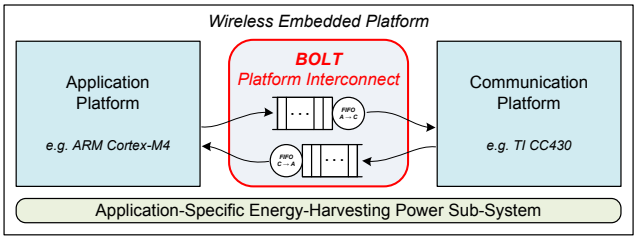


Figure 2: Interconnection of dedicated application and communication platforms using BOLT.

information. This is achieved by passing *messages*, defined as variable-length sequences of bytes up to some fixed maximal length. A platform can share information by writing a message into BOLT. The platform interconnect stores the message in a FIFO queue and notifies the alternate platform of a pending message. The recipient platform may then read out the message from the interconnect when it has the necessary resources to do so.

Message passing through the interconnect is purely asynchronous, since the application and communication platforms are free to operate using their own clock source. As a consequence of this asynchrony, the interconnect is provisioned with two FIFO queues so to enable simultaneous bi-directional message transfer, as illustrated in Fig. 2.

Aside from facilitating *asynchronous message transfer* between application and communication platforms, BOLT exhibits the following unique properties: *(i)* timing predictability, *(ii)* message consistency and *(iii)* persistent message storage, as discussed next.

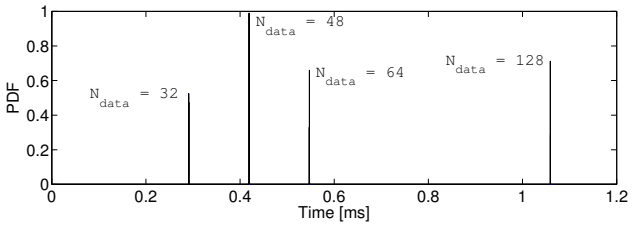
Predictable Timing. Simultaneous and non-blocking read and write operations are guaranteed by design, through the use of independent DMA-enabled SPI peripherals for message transfer coupled with a fully interrupt-driven software state machine. Furthermore, an efficient FIFO queue implementation coupled with cycle-balanced interrupt service routines provides predicable execution time of message operations with bounded jitter.

Message Consistency. The integration of a four-phase asynchronous handshake protocol [1] prevents the addition of partially-written messages into the queue, and the removal of partially-read messages from the queue. Furthermore, the strictly defined handshake protocol ensures that read and write operations are safely committed to memory, *i.e.*, the FIFO queue data structure is updated, before the next operation is allowed to commence.

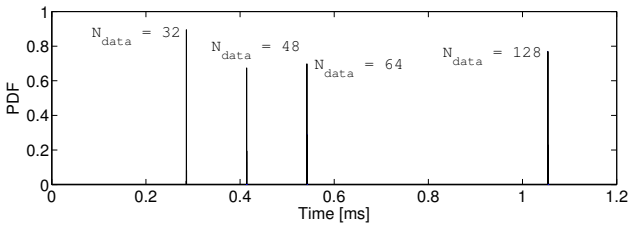
Persistent Message Storage. The two queue data structures and all internal state variables used by the interconnect are stored in non-volatile memory. This ensures that messages successfully written into the interconnect are retained across crash failures, such as power outages. In addition, the persistent storage of messages relaxes the need for application and communication platforms to explicitly acknowledge each message operation using a feedback scheme.

3. EVALUATION RESULTS

Prototype Implementation. A prototype of BOLT has been implemented using the state-of-the-art MSP430FR5969 microcontroller from Texas Instruments. Fig. 1 illustrates the custom development board including all header connectors and debug interfaces. This particular microcontroller offers an ultra-low power dissipation during sleep mode, the availability of built-in non-volatile memory using Ferro-electric Random Access Memory (FRAM), and has an abundance of built-in peripheral support (*e.g.*, multiple SPI interfaces and DMA channels).



(a) Read operation



(b) Write operation

Figure 3: Duration of sequential (a) read and (b) write operations of N_{data} byte messages using BOLT.

Predictable Message Operations. The prototype platform interconnect is evaluated by interfacing two state-of-the-art platforms, specifically, an ARM Cortex-M4 as the application platform, and a TI CC430 as the communication platform. The time to perform sequential read and a write messages was performed more than 4000 times for several message lengths of N_{data} bytes. The empirical probability distribution of the read and write operation times are depicted in Fig. 3. The results indicate a predictable timing behavior, *i.e.*, a tight upper bound on the execution time, with bounded jitter, *i.e.*, a narrow distribution about the mean execution time of less than 1μs.

Negligible Resource Overhead. The additional current drain on the power sub-system incurred by the platform interconnect during periods of no activity, *i.e.*, no messages to read or write, is of primary concern due to its direct impact on the lifetime of the entire platform. The current consumption of the platform interconnect during idle mode is measured at 0.43μA. This current drain is $2.3\times$ lower than the sleep current of the CC430 (*i.e.*, 1μA in LPM4 mode), and $18.6\times$ lower than the sleep current of the ARM Cortex-M4 (*i.e.*, 8μA in stop mode). We therefore conclude that the platform interconnect has a negligible increase on the idle current drain of the combined platform.

Acknowledgments. This work was evaluated by the SNSF and financed by the Swiss Confederation and Nano-Tera.ch.

4. REFERENCES

[1] C. J. Myers. *Asynchronous Circuit Design*. John Wiley & Sons, 2001.

[2] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010.