Radio Frequency Identification Prototyping

ALEX K. JONES, SWAPNA DONTHARAJU, SHENCHIH TUNG, LEO MATS, PETER J. HAWRYLAK, RAYMOND R. HOARE, JAMES T. CAIN, and MARLIN H. MICKLE
University of Pittsburgh

While RFID is starting to become a ubiquitious technology, the variation between different RFID systems still remains high. This paper presents several prototyping environments for different components of radio frequency identification (RFID) tags to demonstrate how many of these components can be standardized for many different purposes. We include two active tag prototypes, one based on a microprocessor and the second based on custom hardware. To program these devices we present a design automation flow that allows RFID transactions to be described in terms of primitives with behavior written in ANSI C code. To save power with active RFID devices we describe a passive transceiver switch called the "burst switch" and demonstrate how this can be used in a system with a microprocessor or custom hardware controller. Finally, we present a full RFID system prototyping environment based on real-time spectrum analysis technology currently deployed at the University of Pittsburgh RFID Center of Excellence. Using our prototyping techniques we show how transactions from multiple standards can be combined and targeted to several microprocessors include the Microchip PIC, Intel StrongARM and XScale, and AD Chips EISC as well as several hardware targets including the Altera Apex, Actel Fusion, Xilinx Coolrunner II, Spartan 3 and Virtex 2, and cell-based ASICs.

Categories and Subject Descriptors: B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids

General Terms: Design, Experimentation, Standardization, Verification

Additional Key Words and Phrases: Design automation, low-power, prototyping, RFID

ACM Reference Format:

Jones, A. K., Dontharaju, S., Tung, S., Mats, L., Hawrylak, P. J., Hoare, R. R., Cain, J. T., and Mickle, M. H. 2008. Radio frequency identification prototyping. ACM Trans. Des. Autom. Electron. Syst. 13, 2, Article 29 (April 2008), 22 pages, DOI = 10.1145/1344418.1344425 http://doi.acm.org/10.1145/1344418.1344425

This work was supported in part by the Technology Collaborative, ADICUS, Inc., the Ben Franklin Technology Development Program of PA, and the University of Pittsburgh.
Author's address: A. K. Jones, University of Pittsburgh, Electrical and Computer Engineering, 3700 O'Hara Street, 348 Benedum Hall, Pittsburgh, PA 15261.

1. INTRODUCTION

Much of the credit for the successful development of current wired and wireless networks is due to the layered concept that is a part of the Open System Interconnection (OSI) Model. While all of the layers may not be used or employed in the manner intended, the idea and thought process were extremely helpful as was true when structures were introduced years ago into the software development process.

Radio Frequency Identification (RFID) is a ubiquitous technology that is exciting, useful, and potentially extremely pervasive. The analogies to networks and software can be easily identified in RFID in that the information network connects to devices termed interrogators or readers that trigger actions and receive information from the unconnected wireless devices termed tags. Thus, there is an information port/connection at the highest level (application layer) and an air interface (physical layer) at the other end of the "stack." However, at this point, the analogy switches more to software in the ability to separate and interface functionality of individual layers as opposed to facilitating peer to peer communication within classical network structures [Ramteke 2001]. Thus, the interrogator/reader is typically a hardware device (with some software) that is a node on an information network and appears as a modem device with the tag functioning as Data Terminal Equipment (DTE) and the interrogator as Data Connection Equipment (DCE).

However, this node (reader) is typically designed, developed, and marketed by an industry where competitive position is based on features, functionality, cost, and reliability of the integrated unit as a whole. As the applications and customer requirements continue to increase and evolve, there is an inherent diversion in each of these features in order to solve a different problem. Thus, essentially each application requires a different reader and infrastructure designed and manufactured to satisfy a particular demand. While this situation may be lucrative to the companies selling legacy readers, it is a detriment to the general advancement of RFID as a technology.

This article presents a series of prototyping environments and tools including both hardware and software components that provide a general procedure to facilitate the development of an open system interface (platform) for RFID interrogator/reader development. The goal is to reduce the disruptive and recurring design and development costs for the infrastructure and multiple air interfaces. Several examples are provided based on current sponsored research and numerous conversations with frustrated end users.

The remainder of this document is organized as follows: Section 2 contains some background on research and related work pertaining to RFID. A design automation flow for generating RFID tag controllers for both microprocessor and custom hardware based systems is described in Section 3 with examples from the ISO 18000 Part 6C standard, formerly called Gen 2 [ISO/IEC 18000-6 2004]. In this section we present two prototyping environments, one for the microprocessor-based tag, and the other for a custom hardware tag. Section 4 describes two prototypes of a power-aware active RFID tag that employs a passive front-end called the "burst-switch." This section describes two platforms,

software and hardware based, to test the burst-switch and an encoding technique for this technology. We present a prototyping technique based on real-time spectrum analysis hardware and software for RFID tags in Section 5. Section 6 presents some results from the use of these prototyping systems for various tag standards and techniques. Finally, conclusions are related in Section 7.

2. BACKGROUND AND RELATED WORK

Research and development in RFID has been focused on hardware and firmware components such as active and passive RFID tags, readers, and embedded software, for the purpose of its deployment in specific application domains. RFID is being incorporated in supply chain management, giving enterprises a real-time view on the location and integrity of their inventories [Sarma et al. 2002]. RFID technology is used in a location sensing prototype system (LANDMARC) for locating objects inside buildings [Ni et al. 2004]. Novel architectures for deployments of RFID in libraries are described in Molnar and Wagner [2004]. RFID tags have been adopted in the Vatican Library in Rome to identify and manage its extensive book and document collection [TI 2004]. RFID tags and intelligent transponders are widespread for vehicle to roadside communications, road tolling and vehicle access control [Blythe 1999]. The medical industry has also deployed RFID technology in "Mobile Healthcare Systems" for positioning and identifying persons and objects inside and outside the hospitals [Li et al. 2004]. Different types of RFID prototype systems are being developed to support all aspects of aviation baggage tracking, sorting and reconciliation, which are surveyed in [Cerino and Walsh 2000]. Recent research has focused on the collection and storage of RFID data using Geo-Time visualization [Shuping and Wright 2005]. Distributed Application Specification Language (DASL) has been used to model and deploy software applications to process the RFID event data [Kaundinya and Syed 2004]. Many of these applications and others would benefit considerably from a tag with long battery life, ideally a life equal to that of the tag. Most of the preceding applications and others use proprietary hardware and software that cannot tolerate changes to the application or standard. However, the use of design automation for the development of flexible RFID systems has not been a topic of research.

In the embedded systems domain, however, the need for meeting aggressive time-to-market requirements has lead to significant research to automate as many design steps as possible. Various design tools and specification methodologies such as Specification and Description Language (SDL), Architecture Description Language (ADL) and Unified Modeling Language (UML) are being used in embedded systems design. SDL [ITU-T 1994] is increasingly being used as a formal, abstract description technique at the system level [Muth et al. 2000]. Optimizations for the performance of SDL-derived system implementations and a tool that supports the complete development process are described in Pereira et al. [2000]. ADL is a language designed to specify architecture templates for Systems on Chips (SoCs). An ADL-based SoC codesign methodology that enables efficient design space exploration of SoC architectures and automatic software toolkit generation has been developed [Halambi et al. 1999].

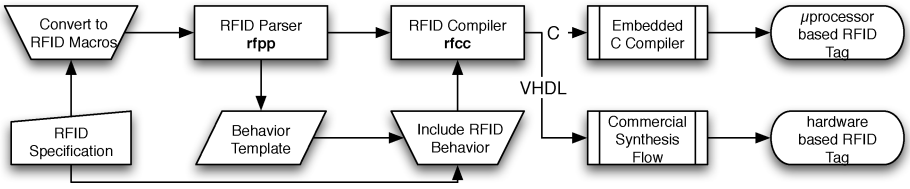


Fig. 1. Specification methodology and compilation flow.

UML is an analogous approach that is a collection of notations for capturing a specification of a software system [Rumbaugh et al. 1998]. UML standards are used for specifying the requirements, documenting the structure, decomposing into objects and defining relationships between objects in a software system. Some tools support code generation from the UML specifications, but are limited by the lack of formalized semantics. Embedded UML is a UML profile for embedded real-time system specification, design, and verification [Martin et al. 2001]. All of the above approaches have enabled the efficient development of embedded systems and mapping from specifications to the implementation models in significantly short times.

Development of a complete RFID system from the specifications in general has required a large amount of design time and expertise, similar to the development of embedded systems. Like embedded systems, RFID systems will benefit from design approaches that can automatically generate target device software. Our contribution to the field of RFID systems design is a tool that can automatically generate the RFID tag controller software based on simple input specifications. This can significantly reduce the design time and the cost of deploying flexible RFID systems.

In addition, the ability to rapidly prototype a custom RFID tag provides an economically viable means for individual companies to provide specialized features to differentiate their products from the competition in that particular application. The tools described here give the company the ability to obtain this differentiation without hiring large teams of skilled personnel.

3. DESIGN AUTOMATION FOR RFID

The RFID communication system consists of a transponder or tag and an interrogator or reader. The format for exchanges between the interrogator and the transponder is a set of commands or primitives that requests that the transponder perform a set of actions. The specifications of these commands vary from one standard to another. The flow of the RFID compiler is specified in Figure 1. This particular compiler can accept virtually any set of commands as input and target a microprocessor or hardware device to provide the RFID tag controller functionality.

3.1 The ISO 18000 Part 6C UHF RFID Standard

Figure 2 shows an example of the ISO 18000 Part 6C protocol for inventory and access of a single RFID tag. In step 1, the interrogator (reader) issues a query. In step 2, the tag responds with a randomly generated 16-bit number.

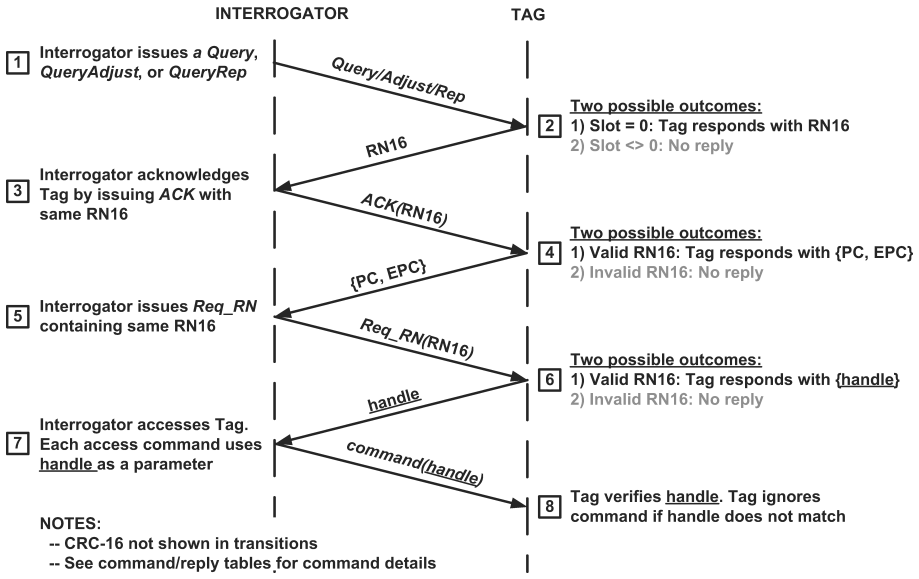


Fig. 2. Example transaction for ISO 18000 Part 6C.

This random number is designed to avoid contention between multiple tags and to ensure that the reader is communicating with only a single tag. The reader acknowledges by returning a random 16-bit number in step 3. This selects only one tag with which to communicate. In step 4, only the tag that issued the matching random number responds with its PC/EPC, essentially its identifier. In step 5, the reader issues a transaction request with the same random number. The tag responds with a transaction handle in step 6. In step 7, the actual transaction is issued with the handle as a parameter. Finally, the tag responds to the transaction in step 8.

An example transaction is shown in Figure 3. The output shown in this figure is generated from a special piece of equipment housed in the University of Pittsburgh RFID Center of Excellence called a real-time spectrum analyzer (RTSA). The example tag uses passive RFID technology requiring the RF energy generated by the reader to be used to power the tag, and to be used for a backscatter-based response. Backscattering uses the reflection of RF energy provided by an external source (in this case the reader) to transmit information. The backscattering device (in this case the tag) either absorbs or reflects the energy of the incoming RF to generate low and high values in the backscattered response.

The prototyping of protocols such as this can now be done in a matter of days by one or two people where otherwise the time and cost are both greater by at least an order of magnitude.

3.2 Specification of Macros

In order to implement the ISO 18000 Part 6C standard into our prototyping environment, we leverage the concept of communication layers introduced in

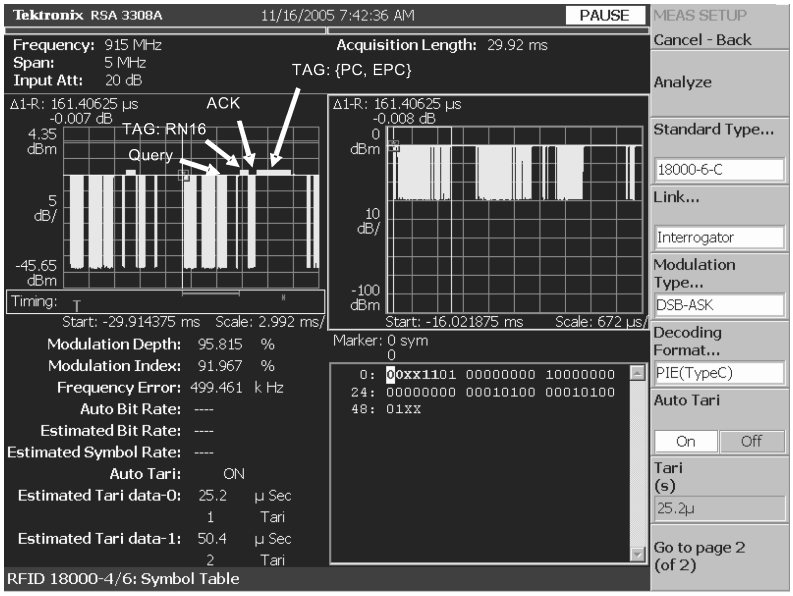


Fig. 3. Real-time spectrum analyzer output of the example ISO 18000 Part 6C transaction.

Section 1. Thus, communication transactions between the RFID reader and the tag can be broken down into a series of *RFID primitives*. To automate the generation of the tag controller for the prototype, we implement these primitives as simple, assembly-like instructions called *RFID macros*. For example, the *RFID macros* required for executing the "write" command of the ISO 18000 Part 6C standard are shown here. The format of the respective fields of each necessary *primitive* and its corresponding response are both illustrated in Figure 4.

The command code of each *RFID primitive* is a unique field or *opcode* that serves as the identifier. Each of the RFID *primitives* also contains a subset of fields with varying lengths providing positions for data present in a command, as can be inferred from Figure 4. Similarly, the tag response to each *RFID primitive* has fields of varying lengths.

Each *RFID macro* description contains a relatively short character string corresponding to the specific name of the *primitive*, a number indicating how many bits are used to represent the *opcode* of this particular primitive as well as the distinct number corresponding to the value of the *opcode*. Additionally, a set of operands that correspond to the *primitive* is included. On the next line, a set of operands is included that corresponds to the standard response.

Figure 5 shows an example *RFID macros* file containing the basic primitives of the ISO 18000 Part 6C standard for initiating a transaction as well as the write primitive. The macros file has a *declarations* section and a *main* section. The *declarations* section allows the user to predeclare the lengths of all of the corresponding fields that occur in each of the *primitives* and responses. In the section identified as *main*, the *primitives* and their specific responses are defined in terms of their fields.

Query Command

QCmd	DR	M	TRext	Sel	Session	Target	Q	CRC-5
4 bits	1 bit	2 bits	1 bit	2 bits	2 bits	1 bit	4 bits	5 bits

Response

RN16
16 bits

Ack Command

ACmd	RN
2 bits	16 bits

Response

PC	EPC	CRC-16
16 bits	96 bits	16 bits

Req_RN Command

Cmd	RN	CRC-16
8 bits	16 bits	16 bits

Response

RN	CRC-16
16 bits	16 bits

Write Command

Cmd	MemBank	WordPtr	Data	RN	CRC-16
8 bits	2bits	8 bits	16 bits	16 bits	16 bits

Response

Header	RN	CRC-16
1 bit	16 bits	16 bits

Fig. 4. Selected primitives and response formats from ISO 18000 Part 6C.

The specific fields can be easily described as illustrated in Figure 5. This provides the user with the ability to adopt any level of granularity in order to manipulate the *primitives* and their corresponding responses. For example, in the *macros* illustrated in the figure, the string denoting the *write* command is `write`. The decimal value of the command code for this specific command is "195," stored using 8-bits.

3.3 RFID Controller Behavior

Communication from the RFID interrogator (reader) is accomplished by transmitting the *primitive* to the RFID tag using a standard air interface. The tag will respond by changing the current state and/or transmitting a designated response message to the interrogator. The user has the capability of specifying the tag behavior in ANSI C.

The details involving size and field position in the command of the interrogator and the corresponding response packet are handled by the compiler. Therefore, complexities encountered in unpacking the command and subsequently packing the response can be abstracted from the user as shown in Figure 6. However, the ability for the user to manipulate each of the individual fields in the response is intact. Therefore, the response customization along with the corresponding state changes can increase in complexity with user ease.

```
declarations
DR(1)
M(2)
TRext(1)
Sel(2)
Session(2)
Target(1)
Q(4)
CRC-5(5)
RN16(16)
RN(16)
PC(16)
EPC(16)
CRC-16(16)
MemBank(2)
WordPtr(8)
Data(16)
Header(1)

main
query(4,4)      DR          M           TRext     Sel     Session     Target   Q      CRC-5
                RN16
ack(2,1)        RN
                PC          EPC         CRC-16
req_rn(8,193)   RN          CRC-16
                RN          CRC-16
write(8,195)    MemBank     WordPtr     Data      RN      CRC-16
                RN          CRC-16
```

Fig. 5. Macros specification.

```
if (current_state != KILLED) {
    if ((current_state == ACKNOWLEDGED —— current_state == OPEN
            —— current_state == SECURED) &&
            ((sel_var == sel) && (target_var == target))) {
        if (session == last_session) {
            if (inventoryFlag == 'A')
                inventoryFlag = 'B';
            else if (inventoryFlag == 'B')
                inventoryFlag = 'A';
        } else {
            slot_counter = rand((1<<Q) - 1);
            if (slot_counter != 0)
                current_state = ARBITRATE;
            else
                current_state = REPLY;
            if (current_state == REPLY)
                RN16 = slot_counter;
        }
    }
}
```

Fig. 6. Tag behavior for `query` command.

3.4 Compiler-Generated RFID Tag Program

The code generation is the final compiler phase determined by the tag behavior and the input macros specification. Code generation can be in the form of ANSI C for general purpose microprocessor controllers or VHDL in the case of hardware controllers. The decode instructions generated by the compiler identify the received *RFID primitive*. For each incoming command, routines are generated by the compiler, that unpack the command generating the fields that it is expected to contain. The corresponding behavior is attached to each field, and the corresponding routines for packing the response are then generated. By virtue of the fact that C is a significant and universally known language compared to hardware description languages, the primitive behaviors are specified in C in the case of a target described with VHDL. Therefore the C code must be converted to a readily synthesizable hardware code.

During the hardware conversion, the C code is converted into a control and data flow graph (CDFG). Compilers commonly use CDFGs to perform optimizations and transformations. Typically, behavioral synthesis tools will also use CDFGs as an internal representation [Tang et al. 2003]. In many cases, the control dependencies present in a CDFG create cycle boundaries during high-level synthesis.

In contrast, in our RFID compiler the CDFG is transformed into an entirely combinational representation by the SuperCISC Compiler. The result is a super data flow graph (SDFG). The SuperCISC compiler [Jones et al. 2005; Hoare et al. 2006] takes advantage of well known compiler transformations including loop unrolling and function inlining and *hardware predication* in order to convert each control dependency into a data dependency creating a combinational representation. The SDFG for the `ionw` is shown in Figure 7. The need for many potentially high-power consuming sequential constructs such as registers and clock trees are removed by this technique. Thus, the resulting SDFG based hardware implementations are extremely power efficient [Jones et al. 2006].

3.5 RFID Compiler Prototype Targets

The RFID compiler provides the opportunity to quickly implement and test different groups of primitives either from RFID standards or proprietary needs of a company. Thus, we have developed two RFID prototype tags to verify the functionality.

We constructed a microprocessor-based prototype. This system is comprised of an Altera Apex FPGA prototyping board used for logic buffer the packets from the air interface, 16-bit EISC microprocessor development board from AD Chips [Cha 2005], and a custom development board created at the University of Pittsburgh for the active air interface.

The prototype shown in Figure 8 utilizes a Spartan 3 FPGA from an Opal Kelly development board as the controller logic and can contain any buffering logic that is required in the tag. For this prototype, the air interface is an off-the-shelf ultra high frequency (UHF) transceiver connected to the FPGA board through a custom board created at the University of Pittsburgh. This board does the analog-to-digital conversion.

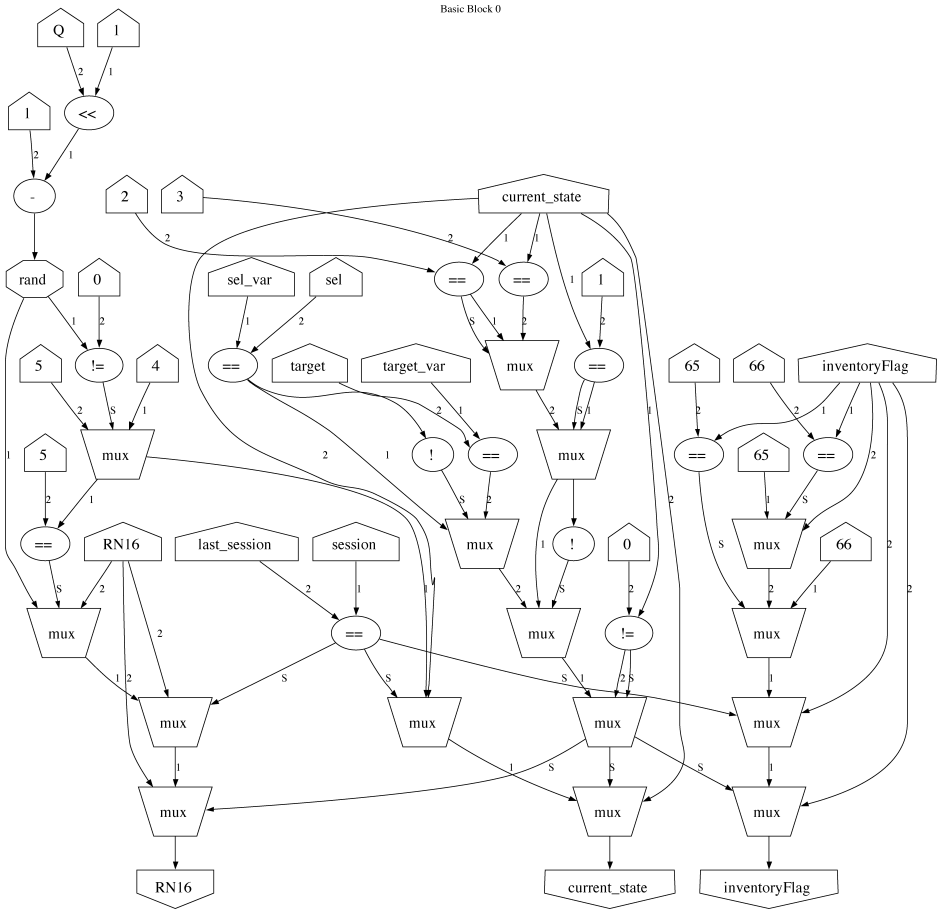


Fig. 7. SDFG for the query command.

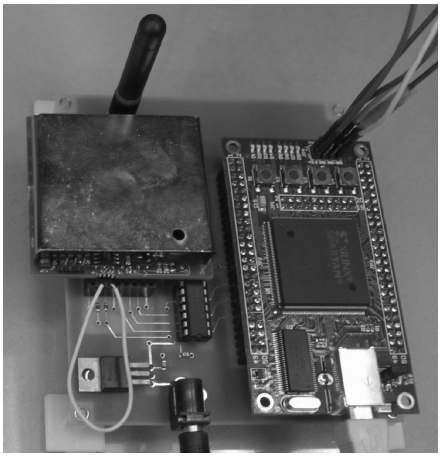


Fig. 8. Spartan 3–based prototype environment for the RFID controller implemented in hardware.

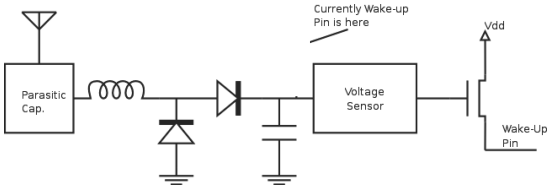


Fig. 9. Burst switch schematic.

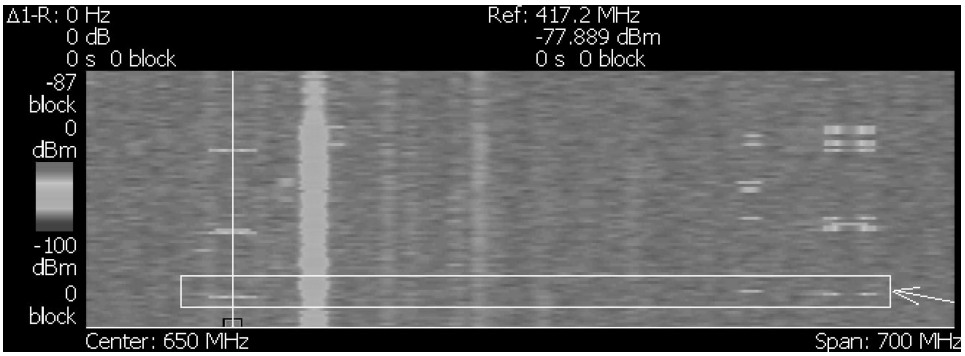


Fig. 10. Burst switch verification with a real-time spectrum analyzer.

4. POWER-AWARE RFID DEVICES

A typical RFID tag consists of three primary elements; (1) transmitter, (2) controller, and (3) receiver. From the classical operational scenario, the transmitter is only needed when there is data to transmit. Thus, the transmitter can easily be turned off when not needed. The controller can be put in a low power/sleep mode to save energy. The only element that is not in a low power mode is the receiver, which is kept active to know when there is an inquiry from the interrogator. Thus, the receiver becomes the determinant of the *shelf life* of an active RFID tag, and it is for this reason that ultra-low power transceivers are critical to the future development of active RFID tags [Akyildiz et al. 2002]. To reduce the power of the RFID tags, we employ a passive transceiver or *burst switch*, which allows the tag to remain in a sleep mode until activated with RF energy.

The burst switch, shown in Figure 9, is a passive receiver that responds to signals transported through the air, which allows the active receiver to be turned completely off. Thus, the battery shelf life is determined primarily by the controller sleep power. The active receiver is turned on only when the burst switch is activated by the wake-up signal(s) from the interrogator. The result is an active tag with minimal power consumption that can normally operate on a single battery for its lifetime.

To verify the burst switch capability, the burst switch was prototyped on a PC Board and tested using a Samsys RFID reader to generate pulses of RF energy to be recognized by the burst switch. The system was tested using a real-time spectrum analyzer (RTSA) from Tektronix, whose output is shown in Figure 10. This test was taken with the reader at a distance of 1 m from the tag for this particular test. In this plot, time is on the Y axis, and frequency is

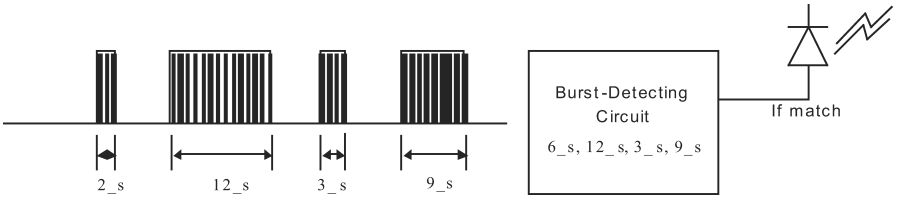


Fig. 11. Example encoding for the burst switch. The wake-up signal contains four bursts of 2, 12, 3, and 9 time units, respectively.

on the X axis. Power is indicated by the color blue being lowest and red being highest. The burst switch activation can be seen in a column near the right at a frequency of approximately 933 MHz. The responses are shown on the left in a column with slightly less intensity at approximately 433 MHz. The white box indicates a burst switch activation that triggered a tag response. The columns of green are radio interference that occur at low intensity within the lab and are not related to the experiment.

The burst switch alone is not sufficient to replace the active receiver for RFID tags. The simple presence of RF energy such as noise or a communication with other wireless systems or even a malicious reader may cause the tag to wake up the active transceiver. To solve this problem we have developed a signal encoding methodology using both hardware and software prototypes [Hawrylak et al. 2006]. The RFID reader generates bursts of energy like those shown in Figure 11. In the example from Figure 11, the reader generates four pulses with lengths of 2, 12, 3, and 9 time units. The tag must detect a unique code from these bursts in order to activate the remainder of the tag. The software based system is implemented with a PIC microprocessor. The hardware based system is designed for implementation in an ASIC or SoC, and is prototyped with an FPGA.

We integrated this encoding scheme into our Spartan 3 FPGA prototype. We tested the system including the new physical layer encoding by connecting a second Spartan 3 board with a Lynx transmitter to transmit the signal of pulses to our Spartan 3 prototype. The received data from the transceiver was output on a pin connected to an oscilloscope to verify functionality, and the result is displayed in Figure 12.

5. RFID HARDWARE AND SOFTWARE SYSTEM PROTOTYPING

In order to test prototype RFID tags and systems, a simulation environment for the system has been constructed and is being explored based on hardware and software from National Instruments (NI) and VI Service Networks [VI Service Networks 2006]. NI has created an FPGA prototyping board based on a Xilinx Virtex II Pro FPGA. The board (NI PCI-5640R IF) also contains a 14-bit ADC/DAC which integrates into the NI RF equipment. The NI RF equipment downconverts the UHF transceivers that operate at 433 and 915 MHz to an intermediate frequency compatible with the FPGA board. The Labview RF visualization software is particularly valuable for testing prototype systems that use RF communication as it shows frequency power spectra over time; see

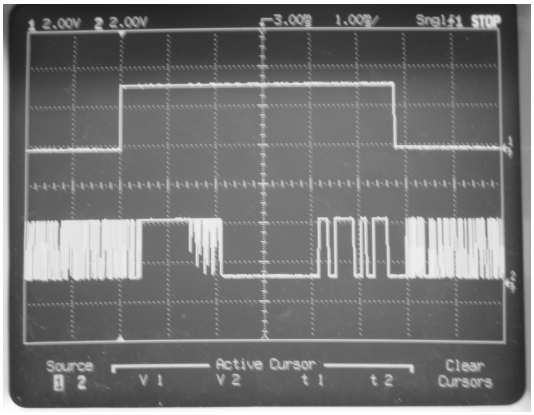


Fig. 12. Detected encoding for bursts of length 2, 12, 3, and 9 time units.

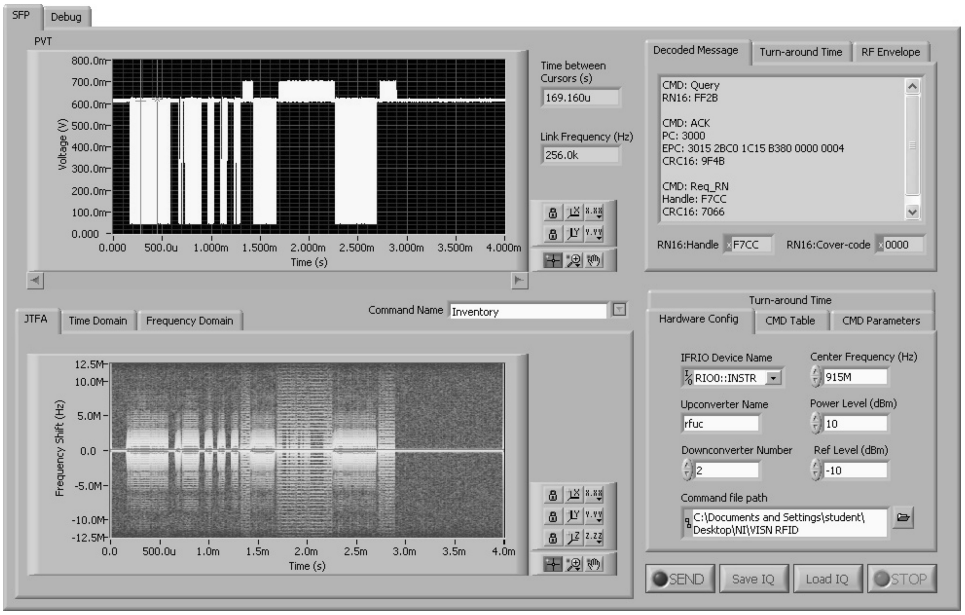


Fig. 13. Labview RF visualization software displaying the gen 2 query command.

Figure 5. Utilizing the FPGA Transceiver board, the simulation environment for RFID systems is configured so that it emulates a real-time communication channel between the reader and the tag.

Typical UHF RFID systems contain a few readers and a large number of tags. Performance of the complete system can be evaluated by varying the coding method, the parameters of building blocks, and the operation distance. However, this prototyping system simplifies the analysis, considering the bidirectional communication of single reader and a single tag. Thus, the physical parameters of this individual communication can be investigated to reveal the parameter values.

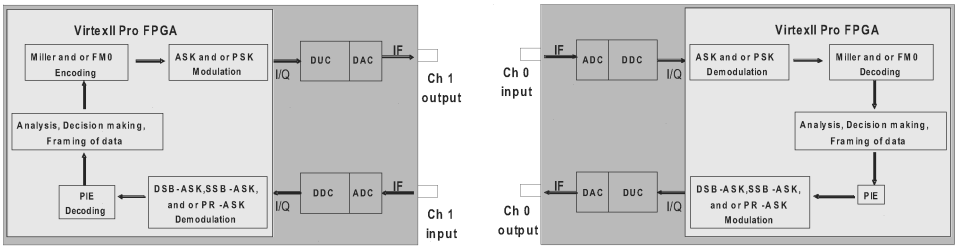


Fig. 14. ISO 18000 Part 6C RFID schematic as implemented in the National Instruments prototyping platform.

For example, in one prototype system the communication link is half-duplex, and is initiated by the reader to the tag followed by the followed by a response from the tag to reader. In a second system using forward link, the reader sends a modulated carrier to the tags which, in turn, powers them up. The tags arbitrate their state and determine which tag responds to the reader. In a third example, return link, the reader sends a continuous wave carrier; the tag receives the carrier for power supply and then backscatters by changing the reflection coefficients of antenna. In this way, the data is transmitted to the reader from the tag.

Figure 14 shows a schematic of an ISO 18000 Part 6C RFID system used as an example in this prototyping system. The figure shows a system model of the forward link and the return link. If a few changes in coding, data rate, reflection method etc. are made, then it will be possible to simulate the system performance under other different conditions. Because of the fact that the NI PCI-5640R IF Transceiver has four channels, two input and two outputs, the proposed system can be implemented on a single board (one FPGA).

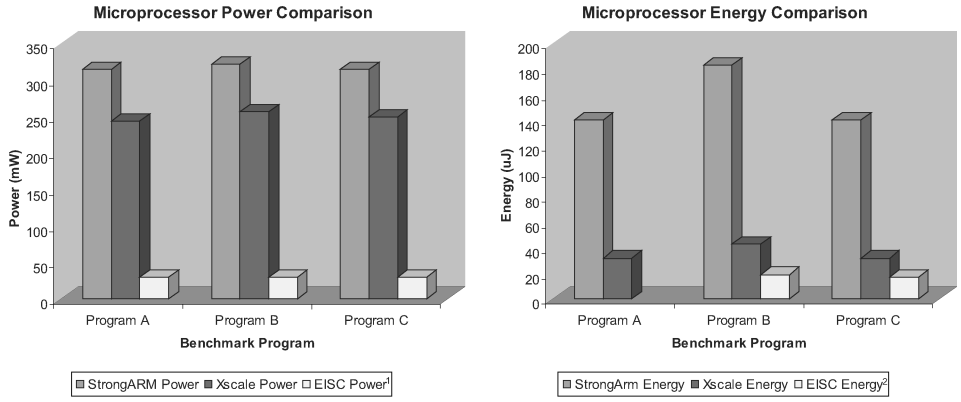
6. RESULTS

The results reported in this paper provide a means of hardware, software, and communications codevelopment that facilitate the simultaneous realization of prototype RFID devices. This technique provides not only functional embodiments but also techniques that optimize power requirements.

6.1 RFID Design Automation

The RFID design automation flow has been used to implement RFID primitives from a variety of different standards such as ISO 18000 Part 7, ANSI NCITS 256, and ISO 18000 Part 6C. The most critical metrics for success with the resulting implementations are area and power of the resulting tag controller. Performance is rarely an issue due to the limited transmission speeds of the RF protocols.

For RFID tags, area is the primary concern as the size of the circuit results in a direct per part cost impact. In many cases, the memory components of the controllers dominate the area impact. In some cases this is dominated by the firmware for a microprocessor-based solution and in others by the memory used for storage in memory enabled RFID tags.



(a) Power. (b) Energy.

Fig. 15. Power and energy comparison of tags with different microprocessor cores.

Power optimization is important in RFID systems as the power supplied to the tags is fixed and battery drain needs to be limited. Because active systems are designed for extremely low-cost large-scale applications, frequent replacement of batteries is not feasible. Current research in power consumption has only focused minimizing power consumption of anti collision protocols [Zhou et al. 2004] and to implement energy conserving access protocols [Chlamtac et al. 1999] in RFID systems. While the transceiver power dominates the power consumption in active tags, our burst switch significantly reduces this impact and the controller power consumption becomes a larger power consideration.

6.1.1 *Microprocessor-Based Tag.* The RFID compiler was used to generate three programs: A with 24 primitives, B with 12 primitives, and C with 4 primitives. Experiments were conducted by executing 1 primitive of Program A, 1 primitive of B, and 1 primitive of C.

The sim-panalyzer [Sim-panalyzer 2000] and XTREM [Gilberto et al. 2004] tools were used to estimate the power dissipation of the microprocessor based tag for the ARM-based cores. Sim-panalyzer is a cycle accurate, architecture level power simulator built on the SimpleScalar simulator. XTREM is a SimpleScalar-based power and performance simulator tailored for Intel XScale micro-architecture. We used SimpleScalar's sim-profile to obtain ARM instruction and instruction class profiles for our software. Because an instruction set simulator was not available for the EISC, the application was run on the development board and the execution time was measured by setting a pin output from low to high upon each iteration, and measuring the duration using an oscilloscope. The energy consumed by the EISC was based on a static power estimate from ADC [Cha 2005], which should be within about 10% accuracy of an instruction level power estimation approach [Russell and Jacome 1998].

Figure 15 shows the power consumption (Figure 15(a)) and energy consumption (Figure 15(b)) of our program on the StrongArm, XScale, and EISC

[1]Static power estimation provided by ADChips [Cha 2005].

[2]Energy calculation is static power consumption multiplied by measured execution time.

Table I.
Area for Implementing the Primitive Logic on a Coolrunner II XC2C512, an Actel Fusion AFS090, and 0.16 μm ASIC. [Macrocells (MCs), product terms (PTs), registers (Regs), function block inputs (FBIs), VersaTiles (VTs). ASIC area is in 100 μm^2

Prims	**2**	**4**	**6**	**8**	**10**	**12**	**15**	**20**	**24**	**30**	**35**	**40**
Xilinx Coolrunner II XC2C512												
MCs	332	335	338	340	350	366	426	447	447	447	449	447
% used	66%	66%	67%	67%	69%	72%	84%	88%	88%	88%	88%	88%
PTs	444	477	514	506	477	552	772	953	993	1106	1181	1213
% used	25%	27%	29%	29%	27%	31%	44%	54%	56%	62%	66%	68%
Regs	262	267	271	271	283	307	422	443	443	443	443	443
% used	52%	53%	53%	53%	56%	60%	83%	87%	87%	87%	87%	87%
FBIs	360	379	408	391	338	396	611	767	801	870	900	914
% used	29%	30%	32%	31%	27%	31%	48%	60%	63%	68%	71%	72%
Actel Fusion AFS090												
VTs	256	258	265	292	300	317	329	371	380	411	434	442
% used	11.1%	11.2%	11.5%	12.7%	13.0%	13.8%	14.3%	16.1%	16.5%	17.8%	18.8%	19.2%
0.16 μm ASIC												
Cells	3809	3836	3841	3835	3859	3961	3990	4140	4170	4235	4264	4298
Area	1.092	1.097	1.098	1.097	1.105	1.116	1.121	1.158	1.161	1.174	1.182	1.187

processors. Both ARM based processors operate in the 250–400 mW range, while the XScale uses significantly less energy. The EISC processor uses an order of magnitude less power, but operates much slower. However, the energy consumed is still less than half of XScale. The energy of executing Program A on the EISC could not be calculated due to the program size exceeding the program memory of the EISC board. This is probably reflective of a real area constraint in an RFID tag, as the chip size would be increased due to memory required for the firmware. Thus, a program size optimization might be necessary for actual implementation.

6.1.2 *Custom Hardware-Based Tag.* Using our RFID compiler, we scaled the number of primitives for the custom hardware-based compiler up to 40 and implemented the hardware in three main targets, a Xilinx Coolrunner II CPLD, an Actel Fusion FPGA, and custom cell-based ASIC hardware at 0.16 μm. Initial results appear in [Jones and Mickle 2006]. It should be noted that in order to fit all 40 primitives, we were required to use the largest Coolrunner II device available (XC2C512) but were able to fit comfortably into the smallest Fusion device available (AFS090).

We present the area and power results for the implementations in Tables I and II, respectively. As previously mentioned, the area required by the custom hardware for ASIC implementation directly impacts the cost. As shown in Table I, the ASIC controller area is quite low, less than 4300 cells for a 40 primitive controller, which is smaller than the 10's of kB of memory required in software-based designs. The three implementations provide levels of power consumption, as shown in Table II. The CPLD power hovers around 1 mW when active, while the FPGA implementation stays between 6 and 9 mW when active. When idle, the quiescent power of the CPLD is 0.05 mW and the FPGA has standby/sleep modes dropping the power to 0.03 mW [Actel 2006]. The direct

Table II.
Power in mW for Implementing the Primitive Logic on a Coolrunner II XC2C512, an Actel Fusion AFS090, and 0.16 μm ASIC

Prims	**2**	**4**	**6**	**8**	**10**	**12**	**15**	**20**	**24**	**30**	**35**	**40**
Xilinx Coolrunner II XC2C512: Quiescent Power 0.05mW												
Total	1.06	1.06	1.06	1.07	1.06	1.06	1.24	1.24	1.24	1.24	1.24	1.24
Actel Fusion AFS090: Quiescent Power 3mW												
Total	6.76	6.82	6.87	7.40	7.41	7.47	7.47	8.23	8.27	8.35	8.36	8.48
0.16 μm ASIC: Quiescent Power <0.4μW												
Total	0.063	0.063	0.063	0.063	0.063	0.063	0.064	0.064	0.064	0.064	0.064	0.065

Table III.
Area for Implementing the Gen-2 Primitive Logic on a 0.16 μm ASIC. ASIC Area is in $100\mu m^2$

Prims	**1**	**2**	**3**	**4**	**5**
Manual	1.1642	1.1933	1.2288	1.2313	1.3212
Automated	1.1326	1.2159	1.2842	1.2942	1.4606
% increase with automation	−2.71%	1.89%	4.51%	5.11%	10.55%

ASIC implementation drops the power consumed to 0.065 mW when operating and 0.0004 mW when idle.

The RFID compiler contains both power and area optimization routines. The power optimizations are described in detail in Jones et al. [2006]. The area optimizations attempt to discover the maximum precision used by signals in the design and propogate that information through the design to reduce the size of storage elements and synthesized functional units. The automatically generated design is expected to be less optimal than a hand design, but provides a reasonable estimate for a system designer to compare different protocols and different implementation targets.

To evaluate the effectiveness of our approach in providing a rapid prototype and accurate estimate of resource requirements of the RFID system the areas of the automated tag designs generated by the RFID Compiler have been compared to the areas of our own manual tag designs for 0.16um ASIC and a Spartan 3 FPGA. The tools used for estimating area are Design Compiler and Precision Synthesis respectively. Table III shows the total area of ISO 18000 Part 6C tag designs for a 0.16 μm ASIC with up to five inventory commands.

Table IV shows the resource utilization of Gen-2 tag designs for a Spartan 3 FPGA. The FPGA resource utilization is almost the same for all the designs and actually decreases slightly with the automated approach. For the ASIC implementation of five primitives, there is a nominal increase in area of up to 10.55%. We note that there is a trend that as primitives are added the area increase percentage rises. This is in part due to how design compiler does resource sharing. We noticed that this did not occur with other synthesis tools for FPGAs. Design compiler does allow resource sharing through use of specialized controls, which provide an opportunity to reduce this overhead.

To understand and compare the complexity of different standards such as the ISO 18000 Part 7 and Part 6C, the RFID compiler has been used to implement

Table IV.
Resource Utilization for Implementing the Gen-2 Primitive Logic on a Spartan 3 FPGA

Prims	**1**	**2**	**3**	**4**	**5**
IOs					
Manual	419	419	419	419	419
Automated	419	419	419	419	419
% increase with automation	0%	0%	0%	0%	0%
Global Buffers					
Manual	2	2	2	2	2
Automated	2	2	2	2	2
% increase with automation	0%	0%	0%	0%	0%
Function Generators					
Manual	720	757	787	789	817
Automated	713	742	814	814	825
% increase with automation	−0.97%	−1.98%	3.43%	3.17%	0.98%
CLB Slices					
Manual	569	572	580	580	588
Automated	559	563	571	571	571
% increase with automation	−1.76%	−1.57%	−1.55%	−1.55%	−2.89%
Dffs or Latches					
Manual	1138	1143	1159	1159	1176
Automated	1118	1125	1141	1141	1141
% increase with automation	−1.76%	−1.57%	−1.55%	−1.55%	−2.98%

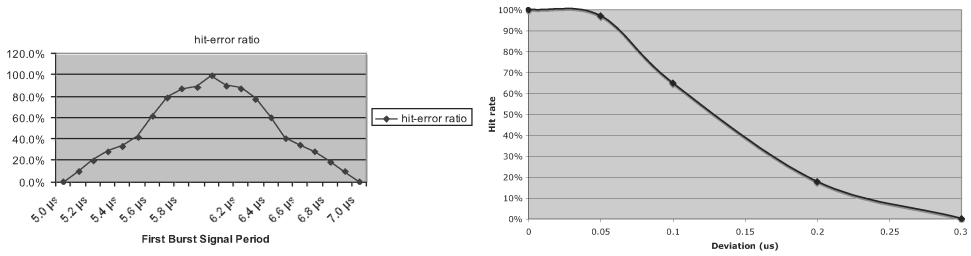
Table V.
Power and Energy Results for Implementing Query, Collection and 10 ISO Part 7 Primitives (inclusive of Collection) as a 0.16 μm ASIC. ASIC Area is in $100\mu m^2$. Dynamic Power is in mW. Quiescent Power $<0.4\mu$W

Primitives	**Dynamic Power (mW)**	**Area** ($100\mu m^2$)
Query	0.06752	1.1642
Collection	0.06308	1.0944
10 primitives	0.06495	1.1232

commands from both of them. A representative command, *Query*, was selected from ISO 18000 Part 6C and has been implemented in hardware and synthesized for the ASIC target. Similarly, the *Collection* command, which realizes similar functionality from ISO 18000 Part 7 standard, was implemented in hardware and targeted to the same ASIC process. Table V shows the power and area results for implementing these two commands. The *Query* command is much larger and higher power consuming than the *Collection* command. We also compared the *Query* command and the *Collection* command with nine additional primitives from the ISO 18000 Part 7 standard. As shown in Table V, the *Query* command is still larger and higher power consuming than these ten primitives from ISO 18000 Part 7.

6.2 Burst Switch Detection

6.2.1 *Software.* The software-based burst switch encoding detector (software detector) was constructed using a ultra low-power Microchip PIC12F635



(a) Single burst 6 μs. (b) Four bursts of 2, 12, 3, 9 μs.

Fig. 16. Hit rate for different time deviations from the detected length burst.

that draws 1 nA when in sleep mode and 100 μA operating at 1 MHz [Microchip 2005]. A Linx transmitter, TXM-433-LR-S is used for the transmitter drawing 5 nA in power down mode, 5.1 mA when transmitting a logic 1, and 1.8 mA when transmitting a logic 0 [Linx 2006]. The total detector current draw when asleep is 6 nA, when receiving a message is 100 μA, and does not exceed 5.1 mA when transmitting.

Consider the transmission of a 96-bit tag ID of an active tag, which is assumed on average to have the same number of 1s and 0s. This is the average case for our NanoTag [Hawrylak et al. 2006]. Thus, the average current draw to transmit a single bit is 3.45 mA. Transmitting at a baud rate of 14400 bits per second with 1 start and stop bit, the transmission battery draw is 0.958 μAhr to transmit the tag ID once per hour. For one interrogation per day, the tag should last over 2000 years on a 1000 mAhr battery. At 10 interrogations per day, the tag will last over 281 years on a 1000 mAhr battery and at 100 interrogations per day, the tag will last over 28 years on a 1000 mAhr battery [Hawrylak et al. 2006]. The actual battery life is dependent on the scenerio in which the tag is deployed.

6.2.2 *Hardware.* To determine the minimum time increment for differentiating a different length pulse, we prototyped the digital portion of the hardware with Spartan 3 FPGAs and connected the generator and detector with a wire. Figure 16(a) shows the percent of reads that detected as 6 μs for values ranging from 5 to 7 μs. In this experiment the deviation was approximately 1 μs. By considering four pulses, this deviation drops to 0% for 0.3 μs as shown in Figure 16(b). When testing with the Lynx transmitter and receiver we found that the clock speed should be reduced significantly below 1 MHz as the resolution of the transceiver is at least an order of magnitude (100 kHz) slower.

The burst switch detector was implemented in hardware using ASIC 0.16 μm technology and examined for power consumption at 10 MHz, 1 MHz, and 100 kHz system clock rates. The number of unique representations for each burst was represented using 2, 4, 8, 16, and 32 bits and for each experiment the number of bursts was held constant at four. The area and power results are shown in Table VI. Even operating at 10 MHz, the power consumption for the largest design is less than 30% of the PIC. However, a 100 kHz clock speed, closer to matching the capability of the transceiver, requires 300X less power

Table VI.
Hardware Burst Switch Detection Circuit Implemented in 0.16 μm Technology at 1.8V

	2-bit	**4-bit**	**8-bit**	**16-bit**	**32-bit**
Area(#cells)	42	52	78	120	205
Area(μm^2)	124	149	235	327	566
10 MHz Clock					
Power(μW)	14.95	17.82	24.33	33.96	53.45
1 MHz Clock					
Power(μW)	1.78	2.21	2.85	3.84	5.86
100 kHz Clock					
Power(μW)	0.17	0.22	0.28	0.38	0.58
Gated Clock					
Power(μW)	0.11	0.17	0.22	0.32	0.55

than the PIC when processing. Interestingly, at such low clock speeds, the clock gated circuit provides little power advantage and the PIC remains much lower power than the ASIC when not receiving a signal.

7. CONCLUSIONS

This article describes several hardware and software prototyping systems and environments for RFID tags and systems. In particular, we have presented a design automation flow that translates RFID transactions or primitives into microprocessor or custom hardware-based RFID tag controllers. We have presented two prototypes, a software-based system using an EISC microprocessor and a custom hardware-based system using a Spartan 3 FPGA. We have demonstrated two "burst switch" related prototypes using both a software and hardware detector for incoming bursts of RF energy to turn on the active transceiver and tag controller. Finally we have demonstrated a full RFID system prototyping environment built from RTSA technology with a RFID FPGA based prototyping board and software built within Labview.

The original problem motivating the reported research for the development of a prototype satisfying a single standard was valued as a $250K effort. The tools reported in this research provide a development suite resulting in an equivalent prototype with a $10K effort. Thus, there is a savings of 96% as a result of using the RFID prototype hardware, software, and communications suite.

These systems provide rapid prototyping solutions to examine the impact of different decisions for building RFID tags and systems. For example, the RFID compiler and design automation flow allows comparison of different configurations of the tag and the impact on area and power for microprocessor or ASIC implementation. A similar comparison is possible for the burst switch detector. The full system testing capability with the RTSA equipment allows similar testing capabilities to standard verification houses such as those for ISO and EPCglobal active and passive RFID specifications and standards.

REFERENCES

ACTEL. 2006. Fusion family of mixed-signal flash FPGAs: DC and power characteristics, 0.5v ed. Actel.

AKYILDIZ, I. F., WEILIAN, S., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. 2002. A servey on sensor networks. *IEEE Comm. 40*, 102–114.

BLYTHE, P. 1999. RFID for road tolling, road-use pricing and vehicle access control. In *Proceedings of the IEE Colloquium on RFID Technology 123*.

CERINO, A. AND WALSH, W. P. 2000. Research and application of radio frequency identification (RFID) technology to enhance aviation security. In *Proceedings of the IEEE National Aerospace and Electronics Conference*.

CHA, Y. 2005. EISC core. ADChips Presentation.

CHLAMTAC, I., PETRIOLI, C., AND REDI, J. 1999. Energy-conserving access protocols for identification networks. *IEEE/ACM Trans. Netw. 7,* 1.

GILBERTO, C., MARTONOSI, M., PENG, J., JU, R., AND LUEH, G. 2004. XTREM: A power simulator for the Intel XScale core. In *Proceedings of the ACM SIGPLAN Joint Conference on Language, Compilers and Tools for Embedded Systems*.

HALAMBI, A., GRUN, P., TOMIYAMA, H., DUTT, N., AND NICOLAU, A. 1999. Automatic software toolkit generation for embedded systems-on-chip. In *Proceedings of the 6th International Conference on VLSI and CAD*. 107–116.

HAWRYLAK, P. J., MATS, L., CAIN, J. T., JONES, A. K., TUNG, S., AND MICKLE, M. 2006. Ultra-low power computing system for wireless devices. *International Review on Computers and Software*.

HOARE, R., JONES, A. K., KUSIC, D., FAZEKAS, J., FOSTER, J., TUNG, S., AND MCCLOUD, M. 2006. Rapid VLIW processor customization for signal processing applications using combinational hardware functions. *EURASIP J. Appl. Sig. Process.* Article ID 46472.

ISO/IEC 18000-6. 2004. Radio frequency identification for item management – part 6: Parameters for air interface communications at 860 mhz to 960 mhz. http://www.iso.org/.

ITU-T. 1994. ITU-T recommendation Z.100: CCITT specification and description language (SDL).

JONES, A. K., HOARE, R., KUSIC, D., FAZEKAS, J., AND FOSTER, J. 2005. An FPGA-based VLIW processor with custom hardware execution. In *Proceedings of the ACM International Symposium on Field-Programmable Gate Arrays (FPGA)*. 107–117.

JONES, A. K., HOARE, R. R., DONTHARAJU, S. R., TUNG, S., SPRANG, R., FAZEKAS, J., CAIN, J. T., AND MICKLE, M. H. 2006. An automated, reconfigurable, low-power RFID tag. In *Proceedings of the Design Automation Conference (DAC)*. 131–136.

JONES, A. K., HOARE, R., KUSIC, D., MEHTA, G., FAZEKAS, J., AND FOSTER, J. 2006. Reducing power while increasing performance with supercisc. *ACM Trans. Embed. Comput. Syst, 5,* 3, 1–29.

KAUNDINYA, M. AND SYED, A. 2004. Modeling event driven applications with a specification language (MEDASL). In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications*.

LI, C., LIU, L., CHEN, S., WU, C. C., HUANG, C., AND CHEN, X. 2004. Mobile healthcare service system using RFID. In *Proceedings of the IEEE International Conference on Networking, Sensing and Control*, vol. 2.

LINX. 2006. LR series transmitter module data guide. Tech. rep.

MARTIN, G., LAVAGNO, L., AND LOUIS-GUERIN, J. 2001. Embedded UML: A merger of real-time UML and co-design. In *Proceedings of the 9th International Symposium on Hardware/Software Codesign*. 23–28.

MICROCHIP. 2005. Pic12f635/pic16f636/639 data sheet. Tech. rep.

MOLNAR, D. AND WAGNER, D. 2004. Privacy: Privacy and security in library RFID: issues, practices, and architectures. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*.

MUTH, A., KOLLOCH, T., MAIER-KOMOR, T., AND FARBER, G. 2000. An evaluation of code generation strategies targeting hardware for the rapid prototyping of SDL specification. In *Proceedings of the 11th International Workshop on Rapid System Prototyping*. 134–139.

NI, L. M., LIU, Y., LAU, Y. C., AND PATIL, A. P. 2004. LANDMARC: indoor location sensing using active RFID. *Wirel. Netw. 10*.

PEREIRA, C. L., SILVA, D. C. D., DUARTE, R. G., FERNANDES, A. O., CANAAN, L. H., COELHO, C. J. N., AND AMBROSIO, L. L. 2000. JADE: An embedded systems specification, code generation and optimization tool. In *Proceedings of the 13th Symposium on Integrated Circuits and Systems Design*. 263–268.

RAMTEKE, T. 2001. *Networks*. Prentice Hall, Upper Saddle River, NJ.

RUMBAUGH, J., JACOBSON, I., AND BOOCH, G. 1998. *The Unified Modeling Language Reference Manual*. Addison-Wesley.

RUSSELL, J. AND JACOME, M. 1998. Software power estimation and optimization for high performance, 32-bit embedded processors. In *Proceedings of the IEEE International Conference on Computer Design*.

SARMA, S. E., WEIS, S. A., AND ENGELS, D. W. 2002. RFID systems, security and privacy implications. http://www.autoidlabs.org/whitepapers/MIT-AUTOID-WH-014.pdf.

SHUPING, D. AND WRIGHT, W. 2005. Geotime visualization of RFID supply chain data. http://www.rfidjournal.com/whitepapers/.

SIM-PANALYZER. 2000. Simplescalar-ARM power modeling project. http://www.eecs.umich.edu/panalyzer.

TANG, X., JIANG, T., JONES, A., AND BANERJEE, P. 2003. Compiler optimizations in the pact hdl behavioral synthesis tool for asics and fpgas. In *Proceedings of the IEEE International SoC Conference (IEEE-SOC)*.

TI. 2004. Texas Instruments' RFID technology streamlines management of vatican library's treasured collections. www.ti.com/tiris/docs/news/news_releases/2004/rel07-07-04.shtml.

VI SERVICE NETWORKS. 2006. VISN RFID solution. `www.vi-china.com.cn`.

ZHOU, F., CHEN, C., JIN, D., HUANG, C., AND MIN, H. 2004. Wireless application drivers for low-power systems: Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems. In *Proceedings of the International Symposium on Low Power Electronics and Design*.

Received September 2006; revised April 2007; accepted November 2007