Reconfiguring the Software Radio to Improve Power, Price, and Portability

ABSTRACT

Most modern software-defined radios are large, expensive, and power-hungry devices, and this hampers their deployment and use in low-power, size-constrained settings like sensor networks and mobile computing. We explore the viability of scaling down the software radio in size, cost, and power, and show that an index card-sized, sub-$150, 'AA' battery-powered system is possible using off-the-shelf components. Key to our approach is that we leverage an integrated, reconfigurable, flash-based FPGA with a hard ARM Cortex-M3 microprocessor which simultaneously enables lower power and tighter hardware/software integration than prior software radios. This architecture allows us to realize the first sub-watt software radio paltform, implement timing-critical MAC protocols, and validate the speculated performance of several recent MAC/PHY primitives and protocols (*e.g.* Backcast, A-MAC, and Glossy) using an IEEE 802.15.4-compliant radio implementation.

1. INTRODUCTION

The advent of the FPGA revolutionized hardware design. It facilitated rapid iteration and prototyping of a diverse array of new architectures. The FPGA has provided these same benefits to the radio space, spawning a new class of devices: software defined radios (SDRs). The primary difference between traditional radios and a software radio is the introduction of an FPGA to replace fixed function components for low-level control and signal processing logic.

Partridge predicts that by 2020, every commercial and military radio will be an SDR system [13]. The military is well on its way with the Joint Tactical Radio System (JTRS [10]). JTRS is the next generation voice and data communication system for the U.S. military and is based on the Software Communications Architecture (SCA [6]), an open-architecture describing how software and hardware works together. Commercial radios have slower to adopt the advantages of SDR systems. GSM basestations are some of the first commercially available SDR solutions [12]. The rapid innovation and changes in radio standards for cell phones places a big burden on the network providers, as hardware has to be regularly upgraded. SDR basestations alleviate this problem as simple firmware upgrades allow the radios to adjust to the changes.

More general SDR hardware architectures offer high performance and are optimized for flexibility [7, 9, 2, 3, 4, 15]. But this has led to large, expensive, and power-hungry systems, ranging from a few Watts to hundreds of Watts of power draw, and costing over $1,000 per radio. This development is contrary to the current explosion of radio devices, which are small, portable, and usually powered from batteries. If Partridge's prediction should become reality, SDR systems must slim down, and start to explore the mobile space.

Mainstream (SRAM-based) FPGAs draw power in four distinct modes. In the powerup mode, the device (look-up tables, interconnect, I/O pads) must be configured, which requires initial charging of the distributed capacitances and causes a significant in-rush current to flow. Proper power sequencing can mitigate this problem, but cannot eliminate it completely. Then, the FPGA enters a configuration mode, which consists of shifting in a several megabits long configuration bitstream. This stage also has a significant current drain and experiences non-trivial delay before the FPGA is ready for use. Finally, during normal operation, the SRAM-based FPGAs dissipate power in two different ways, during active operation and through static leakage. It is this static leakage in SRAM cells that dominates at low activity factors, and makes SRAM-based FPGA ill-suited to a low-power software radio platform.

Since static leakage dominates, traditional approaches to low-power operation become impossible. Frequency scaling, for example, addresses dynamic power so simply lowering or suspending the internal clocks cannot achieve truly low-power operational states on these devices. Voltage scaling, or power gating in the extreme, where the supply is turned off, is not an effective solution either, and may create additional problems as well. First, by turning off power to the device, the memory contents will be lost. Second, the in-rush and reconfiguration currents are incurred on every subsequent powerup following a power down. These startup costs render any savings from the sleep mode moot. Finally, latency incurred due to reconfiguration prevents fast or real-time wake-up techniques that are fundamental in low-power MAC protocols from being used. Therefore, duty cycling is limited to long sleep cycles and cases in which the discarding of the application state can be tolerated.

Platform	P_{Sleep}	P_{Active}	Size	Interconnect	Throughput	Price	Realization
SORA [15]	-	>100 W[c]	36000 cm^{3c}	PCI-Express	16.7 Gb/s	$2000[c]	Research
KUAR [9]	-	>5 W	240 cm^{3f}	PCI-Express	2 Gb/s	-	Research
SODA (180nm) [7]	-	~3 W	26.6 mm^{3e}	DMA[a]	24 Mb/s[b]	-[d]	Simulated
SODA (90nm) [7]	-	~0.5 W	6.7 mm^{3e}	DMA[a]	24 Mb/s[b]	-[d]	Theoretical
WARP [2]	-	10~130 W	800 cm^{3f}	Parallel MGTs[g]	24 Gb/s	$9750	Research
USRP 2[c] [4]	-	7.9~13.8 W[c]	1760 cm^{3c}	Ethernet	1 Gb/s	$1700[c]	Commercial
USRP E100 [3]	5.5 W	9~15 W	1760 cm^3	OMAP 3 GPMC	1.3 Gb/s	$1300	Commercial
μSDR	0.32 W	1.4 W	182 cm^3	AMBA	1.4 Gb/s	$150[h]	Research

[a] Memory bus architecture not specified in [7].
[b] Inferred minimum, may be faster.
[c] Requires a companion PC. Not factored in power, portability, or cost for the USRP 1/2.
[d] SODA is a custom chip that would likely have an extremely high die cost, but low per-unit cost.
[e] Assumes 1*mm* thick.
[f] Assumes 2*cm* thick.
[g] "Multi-Gigabit Transceiver", an interconnect technology built into Xilinx FPGAs. Uses up to 8 parallel 3 Gb/s transceivers
[h] Assumes 1,000 unit production run. See Table 2 for detailed breakdown.

Table 1: A comparison of SDR platforms. The range in power comes from boards whose power usage varies depending on the presence and type of daughter card installed in the system. Where possible a measured idle / sleep power is also shown. For platforms that only list area we make reasonable assumptions on height. μSDR is 10% the cost of the next most expensive SDR platform, yet provides parable speeds in the smallest non-IC package. It uses less power than any realized hardware and nearly ties the previous best theoretical hardware.

2. RELATED WORK

While there are a diverse array of SDR platforms ([2, 7, 9, 15]), none of the current platforms are suitable for low power radio research and development. In particular, μSDR optimizes for power, price, and portability without appreciably sacrificing flexibility or usability. A comparison to previous SDR platforms is shown in Table 1.

2.1 Throughput and Latency

As Schmid identifies in [14], a strongly limiting factor for any SDR system is the available throughput and latency of the interconnect between the "hard" and "soft" portions of the SDR platform. As this latency defines the critical path for the control loop, previous work places significant focus on minimizing it.

In developing SORA, the PCI-Express bus was selected for its bounded latency and high throughput, at the cost of its high power requirements and restricting the platform to the PC form factor. This design affords SORA the support of a complete PC operating system for multi-tasking and control, at the cost of run-time adaptivity. Ettus's USRP platform also relies on the PC operating system for much of its command and control functionality, however it eschews the fixed form-factor of the PCI-E bus, instead relying on more conventional peripheral buses: USB or Ethernet. While these buses do provide high throughput, Nychis et al. find their latency to be highly variable, as high as 9000μs [11]. For the development of low-power protocols, such high and variable latency is untenable.

Instead then, μSDR follows in the footsteps of SODA, tightly coupling the hardware compute engine with an ARM Cortex-M3 core [7]. Both systems allow for low latency (0.46 μs) and high throughput (172 MB/s). Neither SODA nor μSDR use an operating system in the traditional sense, however as a custom micro architecture, SODA goes even further. It defines a slightly customized VLIW+SIMD ISA and sacrifices traditional memory consistency semantics for performance. Users of SODA then must carefully hand-tune their programs not only to optimize performance, but to even run correctly. SODA is not a full-featured SDR platform, rather an advanced DSP architecture. While this microarchitectural compromise allows for better performance per watt, μSDR has been realized in existing commodity hardware (as opposed to just simulation), with a measured full system power draw only slightly higher than SODA's theoretical optimized form, as Table 1 shows.

2.2 Power

We first divide existing SDR platforms into two broad categories: those reliant on a PC or PC-architecture and those that are stand alone. Power and portability are not a design consideration for these PC-platforms (SORA and USRP 2), thus it is unsurprising that μSDR, along with the other embedded platforms, are orders of magnitude better in these metrics.

Of the remaining platforms, μSDR, KUAR, WARP, USRP E100, and SODA, μSDR excels in power draw and is competitive in size with all except the custom chip SODA. Both KUAR and the USRP E100 can be considered "near-PC" platforms that embed low-power, PC-like components in a compact form factor. KUAR does not explicitly report power numbers, but builds a custom board driven by a Pentium M whose most efficient model draws 5 W [5], which we use as a generously conservative approximation for its power draw. The USRP E100 datasheet reports 15 W load with RF daughtercard installed. In our power measurements, shown in more detail in Section 4, we find an idle power draw of 5.5 W. The WARP base system draws about 10 W, but is capable of supplying up to 30 W to each of four daughter cards, allowing for a peak power draw of 130 W.

All of these previous systems have one thing in common: Their power usage is reported in *watts*. With μSDR, we deliver the first realized *sub-watt* SDR platform. This is a significant milestone. μSDR is the first SDR platform capable of running for a full day on a pack of AA batteries.

2.3 Portability

As we enter the era of *sub-watt* SDR platforms, we find that the size, and in turn portability, of the SDR platform becomes a critical design consideration. The current μSDR design is approximately four times the size of popular research motes such as the Mica, TelosB, or Tmote Sky, shown in Figure 2. We note that with the removal of the external memory controller, μSDR's extra interfaces (such as Ethernet), and a slightly more compact layout, the μSDR platform could easily halve its size.

2.4 Price

Perhaps the greatest advantage of the μSDR system compared to previous work is its extremely low cost – an order of magnitude less expensive than other SDR platform. The first large cost reduction comes from building a standalone platform, rather than relying on a costly support PC (SORA, USRP2) or the slightly less costly embedded PC-like environment (KUAR, USRP E100). The next biggest saving comes from eschewing the more traditional daughtercard based SDR approach, instead opting to design a dedicated 2.4 MHz RF frontend. We note that while the μSDR system lacks physical daughtercards, it does not completely sacrifice modularity. Separate RF frontends can be "dropped in" to the schematic and used interchangeably. As evidence, collaborators at other institutions are already designing a new board with a 5 GHz frontend. We also find considerable cost-savings by using fewer and less-powerful FPGAs than the WARP platform. Despite the significantly lower processing power of the μSDR platform, it is sufficiently capable of supporting an 802.15.4 radio, and building the radio on a significantly smaller budget.

3. ARCHITECTURE

This section presents the architecture and design decisions taken to create the μSDR platform. At the heart of our design are three main points: *low-power*, *small size*, and *low cost*. We relax the modular design of existing SDR systems to reduce size and cost, while promoting the FPGA to a first class citizen.

As discussed previously, a fast interconnect is essential to achieve low radio latencies which today's radio standards require. We have to use a fast, but low-power interconnect between the FPGA and the processing core to achieve this. One possibility is to use an external memory interface existing on many low-power microcontrollers to connect to the FPGA. While this allows for a flexible choice of microcontroller and FPGA, it has the disadvantage of greater physical space requirements, limited bandwidth due to a limited bus data width, and potentially increased costs due to additional chip packaging needs.

ARM provides several options both on the low-power computing core and bus architecture front. All major FPGA vendors sell system-on-chip packages that contain high-speed ARM cores integrated with leading-edge FPGAs. The cores and the FPGA are most often connected through the AMBA High-performance Bus (AHB). The AHB is a pipelined, single clock-edge on-chip bus to connect peripherals, memories, and cores on a system-on-chip and provides a bandwidth of up to 16 GBit/s. An even faster version of the AHB is the Advanced eXtensible Interface (AXI) which is used in Xilinx's Zynq reconfigurable SoC that marries a dual-core ARM Cortex-A9 with a Xilinx FPGA in one package [16]. Altera

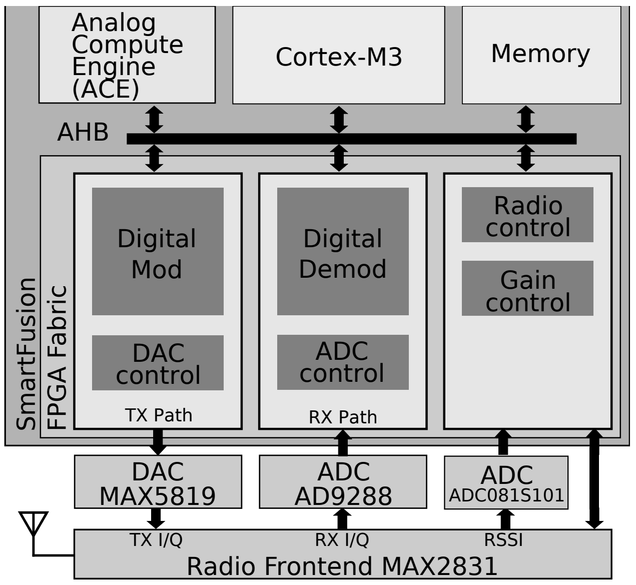


Figure 1: μSDR system architecture. The FPGA Fabric hosts the transmit, receive, and radio control and is directly connected to the AMBA High-performance Bus (AHB), together with the memory, analog compute engine (ACE) and the ARM Cortex-M3 core. This rapid interface allows direct memory-mapped access from the core to the SDR blocks.

has a similar SoC combining a dual-core ARM Cortex-A9 with their Arria V or Cyclone V FPGA series. Altera's bus interface has a peak bandwidth of up to 100 GBit/s [1]. While these systems are very impressive with respect to processing power and speed, their power draw is in the multiple-Watts range, which is outside of our goal for a low-power, battery-operated solution.

Microsemi, a vendor of flash-based FPGAs like the IGLOO series with flash-freeze capabilities [8] offers a third alternative in the SmartFusion customizable System-on-Chip (cSoC). The SmartFusion contains an ARM Cortex-M3 with a flash-based FPGA, connected through an AHB bus interface. The flash-based FPGA brings the advantage of instant-on at power up, as the configuration is stored in the FPGAs floating-gate memory. The AHB interconnect allows a developer to write its own custom peripheral on the FPGA, and access it through memory-mapped IO, as if it were a regular peripheral. This allows for very fast and simple interaction between the software and hardware side of the developed SDR code base, and a flexible division between software-hardware boundary.

Beside the FPGA, the SmartFusion contains a rich set of standard microcontroller peripherals (timers, serial bus interfaces, memories, etc) in addition to an analog compute engine (ACE) with an ADC, DAC, and several amplification and filtering options. By adding an RF radio frontend to the ACE, the SmartFusion would be ready for SDR duties. However, at a maximum speed of only ~700 kS/s, the on-board ADC and DAC would limit us to low data-rate communication.

We add an external high-speed ADC and DAC to the μSDR platform in order to overcome the limits imposed by the ACE. Figure 1 shows a block diagram of the major components on the SmartFusion, as well as the external ADC,

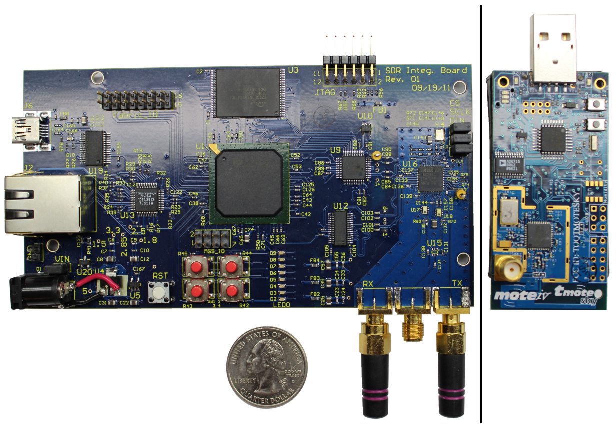


Figure 2: Picture of the μSDR platform. It measures 13 × 7 cm and runs for 7.5 hours without duty cycling from 4 AA batteries, allowing for the first time true mobile SDR experiments. Shown for scaling is the TMote Sky. The current μSDR is approximately four times the size of the Sky. Removing extraneous inputs (Ethernet, etc) and debugging headers, μSDR is approximately twice as large as the Sky.

DAC, and radio frontend chips. Figure 2 shows a picture of the actual platform. The SmartFusion (U1) is the largest chip, located at the center of the PCB. The ADC (U9) and DAC (U12) are located right of the SmartFusion, and the RF frontend occupies the right-hand side of the board, with the RF transceiver (U16) as the center piece.

3.1 Cost Breakdown

Cost was a critical design aspect of the μSDR platform, necessary if one wants to build larger testbeds and deployments of an SDR system. Table 2 shows the cost breakdown of the major μSDR components. Even at moderately small numbers (1k) the μSDR platform should cost less than \$150, including assembly and PCB.

Desc	Part number	Size (mm)	Cost (1k)
FPGA	Microsemi A2F500M3G	17 x 17	\$45
Radio	Maxim MAX2831	7 x 7	\$4
ADC	ADI AD9288	9 x 9	\$4
DAC	Maxim MAX5189	6 x 10	\$5
Discretes			\$34
PCB	6 layers	130 x 70	\$25
Assembly			\$20
Total			\$137

Table 2: Cost breakdown of the μSDR platform.

4. EVALUATION

We evaluate μSDR on its merits as a low-power SDR platform. Specifically, we focus on its power consumption, latency, and hardware flexibility. We further examine the performance of the μSDR system in the exploration of the low-power 802.15.4 MAC protocol.

	SmartFusion	Spartan
voltage rail	static current	static current
3.3V (IOs)	85.6 mA	79 mA
2.5V (Aux)	n/a	88 mA
1.5V (Core)	7.76 mA	n/a
1.2V (Core)	n/a	80 mA
Total Power	294 mW	578 mW

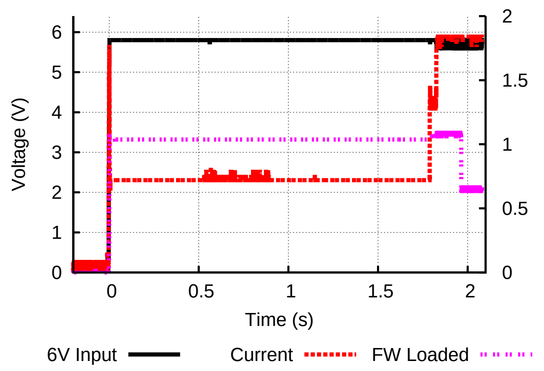
Table 3: Static current at each supply rail for the SmartFusion and In contrast, μSDR has no current spike on bootup and an average current consumption of under 100 mA. To measure time to first useful instruction, we wrote a simple program that simply toggles an IO line, indicating the processor is ready to start executing. We measured the time from power application until the IO asserts. The μSDR requires 259 ms to boot the system, consuming only 0.101 J. Even the latency of μSDR is not ideal for a rapid duty cycled system, which would ideally be in the *microsecond* range as opposed to *milliseconds*. μSDR is remains significantly better than the USRP 2's *seconds* to boot: 7.6× faster in terms of latency and 94.7× lower in terms of energy consumption. platforms.

4.1 Platform Micro-Benchmarks

Designing a low-power system requires building from the ground up with low-power components. In traditional software radio designs, one of the largest power sinks is the FPGA. The recent influx of flash-based FPGAs open the door to low power design. In Table 3 we compare the static power draw of the flash-based SmartFusion to the SRAM-based Xilinx Spartan 3-2000 used in the USRP 2. The superior FPGA core power efficiency of the flash-based FPGA is a direct consequence of the technology. Freed of the requirement to continually refresh the logic cells, the flash FPGA consumes very little static power. As a young technology, we expect these innovations in static power for flash-based FPGAs to improve rapidly, as evidenced by the Microsemi IGLOO series, whose Flash*Freeze technology lowers the FPGA static power draw as far as 2 uW [8].

In addition to the impact on static power usage, flash-based FPGAs also excel during cold-boot. For an FPGA to be useful, it must at some point be configured. In the case of SRAM-based FPGAs, this configuration must occur at every boot, as the configuration of the FPGA is lost when power is no longer supplied. This configuration period introduces a non-trivial delay for the FPGA wake-up process as well as consuming a relatively large amount of power. Most importantly, the latency imposed by the configuration window inhibits effective duty-cycling of the FPGA. In Figure 3, we compare the cold boot times of the SRAM-based USRP 2 (3(a)) to the μSDR (3(b)). After a system in-rush current spike of 1.8 A, the USRP 2 averages a 700 mA current draw for 1.964 s while it configures its FPGA. This long configuration process and high current consumption consume 9.56 J of energy, a cost that must be paid every time the FPGA is power cycled.

Given that the 259 ms boot time is too highly latent to support aggressive duty-cycling, μSDR also presents a sleep mode as a median option between system full on and full off. Unfortunately, the SmartFusion chip is only capa-



(a) USRP2 Cold Boot

(b) μSDR Cold Boot

Figure 3: USRP2 and μSDR cold boot comparison. Both systems power on at *0 s*. As expected, the USRP2 (a) has a current spike of 1.8 A, and exhibits an average current draw of 700 mA at 6 V. It takes 1.96 s to load the firmware from an external SD card, resulting in 9.56 Joules of energy consumption during cold boot. In contrast, the flash-based μSDR doesn't have an in-rush current and the average current is less than 100 mA at 3.3 V. Moreover, μSDR requires 259 ms to boot the system, resulting in only 0.101 Joules of energy consumption. μSDR is 7.6× faster and uses 100× less energy than the USRP2 to start-up.

ble of sleeping the M3 core (via WFI[1]). Fully sleeping the FPGA would require technology similar to the Flash*Freeze mode from the Microsemi IGLOO line of low-power FPGAs. Unfortunately these FPGAs do not yet support the tight integration with a hard CPU and are currently unsuitable for the μSDR platform. Until technology similar to Flash*Freeze comes to the SmartFusion, μSDR is still capable of lowering the FPGA power via frequency scaling. In μSDR's frequency scaling mode we turn off the external crystal oscillator and used the FPGA's internal RC network instead. The frequency of the RC network is reduced to 6.5% of the original frequency (to 3.125 MHz). We additionally turned off the PLL and bypassed two other clock sources for FPGA logic to further reduce the system current. Table 4 examines the impact of these power saving efforts. Simply placing the processor in WFI state and power gating all external components reduces system power consumption to 108 mA with only 34 μs of wakeup latency. Introducing FPGA frequency scaling reduces system power further to 67 mA, but imposes a 3.01 ms latency to turn on the crystal oscillator and stabilize the PLL until the processor can be woken.

[1]WFI: Wait For Interrupt, and ARM instruction that places the core in a lower power sleep state until interrupted. The contents of registers are preserved but the rest of the core is power gated

Frequency Scaling	Wake-up Latency	System Power
off	34 μs	518 mW
on	3.01 ms	322 mW

Table 4: Sleep power and latency trade-off. Without frequency scaling, μSDR draws 108 mA while Waiting For an Interrupt *WFI*. It wakes-up from this state in only 34μs. With frequency scaling, μSDR saves 38% in power while taking 89× longer to wake up.

Spartan 3-2000. The Spartan is a more traditional SRAM-based FGPA whereas the SmartFusion is a flash-based design. Measurements are taken after both devices have been configured and allowed to settle. We observe similar static current at the IO buffers, but significant differences at the core supply. The SmartFusion current is 10× less than the Spartan. As a consequence, the SmartFusion requires approximately half as much static power as the Spartan, making flash-based FPGAs a much better desgin decision for low-power

We finally compare μSDR to the USRP E100, USRP's "embedded" series. The E100 is built on the GumStix platform, a small ARM-based PC-like device. The advantage of the E100 is it requires no controlling computer, rather it is built in to the system. As a consequence, however, we find in Figure 4 that the E100 actually has greater power draw that it's PC-encumbered counterpart.

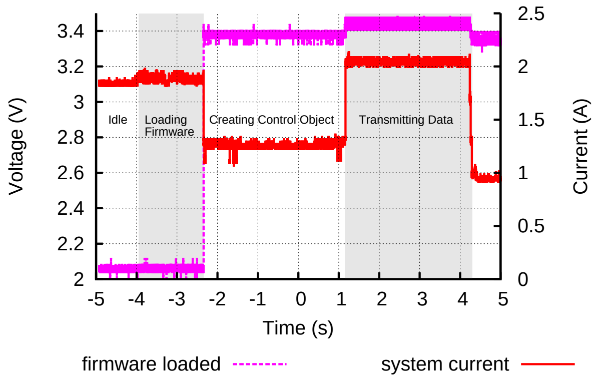


Figure 4: USRP E100 power profile. E100 runs a full embedded Linux platform, making cold-boot an untenable comparison. Rather we show the timing from an idle, booted system to load a firmware image (configure the FPGA) and send the data. The firmware loaded signal is recorded from a firmware loaded pin. After configuration, the system idle power drops to just under 1 A at 6 V.

5. DISCUSSION

This section looks at the radio requirements claimed in prior work and comparing the current μSDR to the performance it is supposed to achieve in order to compete with current commercially available low-power wireless radios.

5.1 Radio Duty-Cycling

Typical duty-cycled radio platforms achieve <1 mW sleep power, wake up in μs, and draw tens to hundreds of mW while actively using the radio. μSDR draws in it's lowest sleep state 322 mW (wakeup in 3.01 ms) or 518 mW (wakeup in 34 μs). During transmit, μSDR draws 1.4 W, and 1 W while receiving. While the sleep current is still three orders of magnitude greater than typical duty-cycled systems, it is a good start to explore the space of battery powered SDR platforms since nodes can last for many hours on 'AA' batteries.

5.2 Low-Power FPGA

During sleep mode, radio frontend, ADC, and DAC support sleep modes in the μW range. Still, μSDR draws hundreds of mW. Unfortunately, the SmartFusion does not include the latest Flash*Freeze technology, allowing to clock-gate the FPGA itself. Thus, even in deep sleep, the FPGA is still running and wasting energy. We expect that future iterations of the SmartFusion will include such technology.

6. CONCLUSIONS

Software-defined radios are reconfigurable communication systems that transcend historical boundaries between hardware and software subsystems, physical and logical layers, and analog and digital domains. In so doing, they enable radical new architectures, novel radio designs, and high-performance protocols that are not easy to design, implement, or evaluate using traditionally-layered approaches. Although modern SDR platforms have been used to explore many facets of the wireless design space, their current architectures make it very difficult to explore the *low-power* design space. Their use of SRAM-based FPGAs result in high static and dynamic power draws, their slow startup times are not amenable to rapid duty cycling, their radio front-ends do not support power controls, and their processing requirements place a heavy load on the system. As a result, fertile application areas like mobile phones and sensor networks that could benefit from radical approaches, but which require low-power operation, remain relatively unexplored.

We developed μSDR to address this inequity. μSDR is a small, low-cost, and low-power software-defined radio platform that leverages emerging technology like highly-integrated radio front-ends and mixed-signal FPGA processing back-ends. This paper demonstrates that a software radio with a footprint of below 100 cm^2 and costing less than \$150 is able to operate from a set of 'AA' batteries, and that we can expect a battery-life similar to today's smart phones. This work enables new research areas that were completely out-of-reach or existed only in severely limited forms in low-power nodes. Hence, μSDR is an enabling technology for many high-impact, large scale applications of low-power, ad-hoc wireless networking where high performance and/or precise timing are required, including full-duplex wireless communication, synchronous concurrent communication, high-frequency power metering, infrastructure less audio/video streaming, and structural health monitoring.

7. REFERENCES

[1] ALTERA. Cyclone V SoC FPGA Hard Processor System.
[2] K. Amiri, Y. Sun, P. Murphy, C. Hunter, J. Cavallaro, and A. Sabharwal. Warp, a unified wireless network testbed for education and research. In *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, pages 53 –54, june 2007.
[3] Ettus Research. USRP E100.
[4] Ettus Research. USRP N200.
[5] Intel. Intel pentium mobile processor, Apr. 2012.
[6] JTRS Standards. SOFTWARE COMMUNICATIONS ARCHITECTURE SPECIFICATION.
[7] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner. Soda: A low-power architecture for software radio. In *Proceedings of the 33rd annual international symposium on Computer Architecture*, ISCA '06, pages 89–101, Washington, DC, USA, 2006. IEEE Computer Society.
[8] Microsemi. Microsemi IGLOO Series.
[9] G. Minden, J. Evans, L. Searl, D. DePardo, V. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill, A. Wyglinski, and A. Agah. Kuar: A flexible software-defined radio development platform. In *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, pages 428 –439, april 2007.

[10] J. Mitola III. SDR architecture refinement for JTRS. In *MILCOM 2000. 21st Century Military Communications Conference Proceedings*, volume 1, pages 214–218. IEEE, 2000.

[11] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste. Enabling mac protocol implementations on software-defined radios. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 91–105, Berkeley, CA, USA, 2009. USENIX Association.

[12] Octasic Inc. Vocallo BTS.

[13] C. Patridge. Realizing the future of wireless data communications. *COMMUNICATIONS OF THE ACM*, 54(9):62–68, September 2011.

[14] T. Schmid, O. Sekkat, and M. B. Srivastava. An experimental study of network performance impact of increased latency in software defined radios. In *Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, WinTECH '07, pages 59–66, New York, NY, USA, 2007. ACM.

[15] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker. Sora: high-performance software radio using general-purpose multi-core processors. *Commun. ACM*, 54(1):99–107, Jan. 2011.

[16] Xilinx. XILINX ZYNQ-7000 EPP.