A Case for Custom Silicon in Enabling Low-Cost Information Technology for Developing Regions

Z. Foo, D. Devecsery, T. Schmid, N. Clark, R. Frank, M. Ghaed, Y. Kuo, I. Lee, Y. Park, Z. Renner, N. Slottow, V. Vinay, M. Wieckowski, D. Yoon, C. Schmidt†, D. Blaauw, P. Chen, and P. Dutta

University of Michigan
Ann Arbor, MI 48109

†Literacy Bridge
Seattle, WA 98101

ABSTRACT

Information and communications technology has the potential for deep social impact in developing regions but today's typical ICT devices – laptops, mobile phones, and similar devices – are often still too expensive for many scenarios. In this paper, we argue that custom integrated circuits can enable a new tier of low-cost information access devices with a price point that will make them widely accessible. And, with control over the silicon, these systems can economically address many other challenges. To evaluate our claim, we focus on a deceptively simple problem – low-cost information access for illiterate populations through audio recordings – and show how custom silicon allows us to reduce cost, lower power, leverage conventional infrastructure in unconventional ways, and optimize the interface for usability. In particular, we show how a *rural audio computer* can be designed around just three chips, use an inexpensive capacitive touch interface, employ inductive communications for peer-to-peer data transfer, and employ content download over GSM voice and FM broadcast as two wide area options. The resulting design point – enabled by aggressive silicon integration – affords a device that can be built for $7.77, a third of the cost achievable using commercial off-the-shelf components.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication, store and forward networks*; C.3 [**Special Purpose and Application-Based Systems**]: Microprocessor/microcomputer applications; H.4 [**Information Systems Applications**]: Miscellaneous; H.5 [**Information Interfaces and Presentation**]: Miscellaneous

General Terms

Design, Human Factors, Performance, Reliability

Keywords

Developing Regions, Data Dissemination, Talking Books

ACM DEV'10, December 17–18, 2010, London, United Kingdom.

1. INTRODUCTION

Recent studies have illustrated the benefits of timely, actionable information on rural agrarian populations in developing regions. In the *Avaaj Otalo* project [12], a mobile phone-based, interactive voice forum enabled the "emergence of norms, persistent moderation, and a desire for...interaction with...authorities and open discussion with peers" in India. Users of the system found value in listening to questions and answers with 77% of interviewees citing this as the main reason they liked the forum. The Digital Green project [7], "seeks to disseminate targeted agricultural information to small and marginal farmers in India using digital video." A four month trial of the system showed a six to seven times increase in adoption of certain agricultural practices. The *Talking Book* project [14] deployed a custom-built audio computer in rural Ghana to study "the impact of giving on-demand access to guidance created by local experts" and found that in a village with 90% illiteracy and no electricity, Talking Book users significantly increased crop production over farmers who did not use the devices.

Encouraged by results illustrating the benefits of timely, actionable information on rural agrarian populations in developing regions, this paper explores the question of how to scale, and make economically viable and sustain, such technologies for the nearly 1.5 billion people who live without electricity, the 752 million who are illiterate, the great majority of whom do not own mobile phones and live in rural areas with limited or no data connectivity [11, 17].

Of all the challenges – cost, power, connectivity, illiteracy, and usability – we argue that cost is the least flexible challenge facing timely access to actionable information. If cost were not a factor, one could conceivably provide audio information to illiterate users using audio recordings on a smart mobile phone and simply recharge batteries using a solar panel or generator. One could use GPRS or EDGE where available for content downloading or use an integrated or external FM receiver to download content where GPRS or EDGE was not available, share content using WiFi in ad hoc mode, and offer graphical or voice-activated user interaction.

That upfront cost is the dominant factor in widespread, sustained deployment is illustrated by two first-hand observations from a recent trip to remote villages in Ghana by members of our research team. First, these subsistence farmers (who grow enough food for their families with little or no surplus for cash income) tend to purchase a low-quality ax that costs $10, but usually breaks quickly and needs to be replaced, rather than spend money on a high-quality ax that costs $20 but will last a life time. Second, the majority of electronics owned by these populations, again based on our observations, typically cost less than $10 and are almost never more than $20, establishing the viable price points for typical consumers: *economic viability requires a target price of approximately USD $10.*

In Section 2, we outline why existing devices, most notably mobile phones, have been unsuccessful in meeting this need. In particular, we show how smart phones fall significantly beyond the required cost bracket. Even inexpensive basic phones do not quite address the needs of this population. Basic phones, while approaching the required cost point, lack the data storage, offer no simple data transfer methods, and pose logistical issues in that they require "reflashing" (an operation highly-specific to each phone model). In addition, mobile phones have high power draw and poor speaker efficiency, which makes their ongoing cost of ownership significantly higher than the devices currently owned by these populations.

Given the need for a custom device, in Section 3 we present the *rural audio computer*, a device tailored to the unique ICT needs of rural developing regions, and the system architecture that supports its operation. We then argue in Section 4 that custom silicon offers a way to cut the Gordian Knot that ties cost, power, connectivity, and usability. A custom chip, through: (i) *component integration*, (ii) *reduced power draw*, (iii) *application-specific design optimizations*, and (iv) *electronic-mechanical co-design* offers dramatic cost reductions that can make information technology widely accessible. We expand on these claims in the next few paragraphs.

First, component integration reduces chip count and pin count, both of which reduce cost. Current systems constructed from commercial off-the-shelf components (COTS) can consist of as many as 150 discrete components and six major chips [14]. This drives up cost through margin on each individual device and packaging due to high pin-out counts (which can be as high as 50% of the chip cost). By developing custom silicon, we show that the entire system can be integrated in a handful of devices. This saves costs directly through fewer devices and reduced packaging, but also indirectly by allowing a much smaller and simple PCB design (consisting of just 1 or 2 layers). Other indirect benefits also occur, such as lower cost of plastic packaging and reduced shipping costs resulting directly from a smaller PCB footprint.

Second, custom design provides extensive opportunity for a reduction in power draw (overall 8$\times$ in the present case study) which reduces total cost of ownership (TCO) by increasing the lifetime on a pair of (disposable) batteries. Again, indirect benefits of lower power exist, such as smaller casing requirements due to smaller or fewer batteries leading to lower fabrication and shipping costs, which accrues to both lower upfront costs and lower operating costs.

Third, custom integration allows for a design specifically targeted at the developing regions market. Carbon-zinc batteries, for instance, are almost the sole battery option for the target population. These batteries have significantly different discharge characteristics from alkaline batteries common in the West. The choice of battery chemistry, regulator architecture, conversion efficiency, and other factors all play into the design of a low-cost regulator. A custom converter better accommodates carbon-zinc discharge curves.

Finally, judiciously designed silicon not only reduces the number of electronic components, but also eliminates other mechanical components. For instance, custom silicon makes capacitive touch sensors economical which eliminates the need for membrane push button switches, which in turn increases reliability and decreases cost. Similarly, the coil necessary for inductive coupling-based, near-field communication (NFC) can be printed on the PCB, eliminating the need for a discrete antenna and additional radio chips.

Section 5 discusses some software implications of the hardware design and usage scenarios, including the application programming environment, a robust software flash translation layer, and the cache and memory management. Section 6 presents our content distribution network for dissemination over FM and GSM to audio computers, and locally among the devices using NFC.

2. BACKGROUND

System designers using ICT targeted at developing regions face many different challenges than those that apply to more affluent markets. In this section, we outline some of these challenges, discuss contemporary mechanisms for information access, and highlight the state-of-the-art in low-cost digital talking books.

2.1 Practical Challenges in Rural Settings

The world's poorest 2.6 billion people live on $2 per day, or less, and make up approximately 40% of the earth's population. For people designing ICT products aimed at this population, cost is key. One must consider both the cash-flow challenges associated with initial purchase and the overall return on investment, which factors in all ongoing costs. The TCO includes the initial cost, the cost of power, and the cost of information access using the product. For instance, a basic mobile handset may cost between USD $20-$40, but listening to new content costs an additional $0.10-$0.30 per minute throughout most of Sub-Saharan Africa [3]. Given our observation that the target users often prefer to listen to a recording several times, this recurring cost could quickly dominate the TCO.

Radio is widely used throughout rural developing regions, but it does not allow users to find the information they need, when they need it; nor, does it allow users to listen to a message several times.

Cassette players, with their electro-mechanical action, are not energy efficient and they add significant costs for content reproduction, distribution, and sharing with more than one device.

In contrast, a smart phone that can download, store, and recall an audio recording allows users to replay content as necessary for only the cost of power, but these phones require an initial investment that is just not possible for most of the poorest 40% of the world. Similarly, designing a device to include a built-in renewable power source will reduce TCO in the long-run through power cost reduction, but the additional cash required to acquire the device may put it out of reach for this audience. Mid-range phones with FM radios and recording capability are still too expensive for one billion people living in extreme poverty; but more importantly, they are not designed as learning devices for people raised in oral cultures with little formal education. The human-computer interface needs of these users is significantly different from literate users [15].

Only 15% of rural households in Sub-Saharan Africa have access to grid electricity [1]. Worldwide, 1.5 billion people have no access to electricity and, what's worse, is that the situation is not expected to improve much in the next 20 years: by 2030, the International Energy Agency predicts the number to drop from 1.5 to 1.3 billion people. Therefore, any ICT product designed for the poorest people in the world cannot depend on grid electricity. Some will walk two or three hours to recharge their phone in a nearby town; but, this scenario works best when the phone is used for infrequent and brief calls. In fact, phone usage may be infrequent by necessity when network reception is too weak or not available in ones village (as applies to millions of mobile subscribers). In each of these scenarios, a successful example of using mobile phones for communication does not necessarily translate into the ability to use the same tool for ongoing learning and on-demand reference.

While basic phones and networks in the most impoverished areas often include SMS text capabilities, there is a significant barrier for most potential users: illiteracy. Across South Asia and Sub-Saharan Africa, self-reported adult literacy rates are approximately 60% (these numbers are typically self-reported and have been shown to significantly underestimate the problem when follow-up testing occurs). In many villages, the literacy rate may be as low as five to ten percent. Any ICT solution depending on text is not an option for the people who have the greatest need.

Some mobile phone-based ICT solutions have adopted interactive voice response (IVR) systems. The main challenge with such a system is usability, particularly on the input side (user selects an option). Those systems applying voice recognition are limited to the largest languages, which again excludes most of the potential beneficiaries of such a system. Those systems using keyed input face a usability challenge, particularly for people without numeracy skills or without exposure to any ICT beyond a radio. On the output side (prompts and status messages), graphical icons increase device cost and exclude those with a visual disability; visual disability is twice as common in developing countries as developed countries – and yet Braille is much less available.

2.2 Contemporary Dissemination Vehicles

Today, the most efficient way to teach practices for reducing child mortality rates is often by driving a pickup truck over rough unpaved roads to a village that hasn't been visited recently. Upon arrival in a targeted village, the local health expert might gather anyone who is available and overwhelm them with numerous messages about disease prevention. Unfortunately, those who were able to attend will have difficulty recalling any specific guidance that they did not immediately apply – they are not able to replay the guidance on demand nor can they take notes due to illiteracy.

Low-cost rural audio computers, or talking books, can serve as on-demand audio libraries of local content, but broad content distribution remains a challenge. Broadcast is ideal for wide-spread audio dissemination but it comes at the disadvantage of being a one-way channel from the content distributor to the talking Book users, for example. There are two ways a user can receive specific content directly from the distributor. By giving the distributor feedback, the broadcast could include requested content during a specific schedule. Or, the user could use a unicast channel to the distributor. Both solutions have different requirements on the communication channel. Sending feedback to the content distributor requires sending small messages containing topics of interest. The content distributor can then use this feedback to create the broadcast programming.

SMS messages, while expensive, are universally accessible. In Africa, one SMS costs between $0.05 to $0.21 [4]. While this is comparable to a 1 minute voice call [3], feedback is short and small compared to the data volume necessary to receive audio data. Thus, SMS can provide a high-cost, low-bandwidth feedback channel.

Another possibility of retrieving specific content on the phone is through the GSM data channel. The General Packet Radio Service (GPRS) or Enhanced Data rates for Global Evolution (EDGE) allow direct download of digital audio data over the Internet to a phone. However, these services are not available everywhere, and require an additional data plan. Access to the data modem of the phone is another issue that a talking book would have to overcome: not all phones expose this capability to external devices, and there is no standard modem port on cellular phones.

A further possibility for direct content dissemination is through the GSM voice channel. While recent work provided methods for transmitting digital data over the GSM voice channel [6, 10], the needs for a talking book are different. The GSM voice channel already provides excellent voice compression algorithms. Thus, if we connect the Talking Book headset port to the phone headset port, we can directly record the audio from the GSM channel. The user dials a regular phone number that was potentially provided during the audio broadcast, or that is well known to the user community. A regular phone menu system would allow the user to select specific content. Thanks to SIP, and the open source software Asterisk [2] such a private branch exchange (PBX) system would be inexpensive to construct and maintain.

Figure 1: The Talking Book: a state-of-the-art rural audio computer designed specifically for on-demand information access in developing regions. To access audio content, users navigate through voice prompts using the small number of orange input buttons visible on the device.

However, we would still need to provide digital meta information on the current voice recording, disseminate lossless software updates, and provide feedback from the book back to the distributors. The requirements for this type of communication are easy to encode and decode on the book and low frequency spectrum needs as it has to be transmitted together with the audio. A bandpass filter could then remove the modulated signal from the voice stream, before the recording gets stored as audio on the book.

2.3 Modern Digital Talking Book

The Talking Book [14] represents a promising point in the design space from a utility and usability perspective. The device allows users to play, record, and categorize audio recordings and to copy those recordings directly to any other Talking Book. The audio system instructions are easy to localize and each device can include multiple system languages. The device also supports programmable interactive applications such as multiple-choice quizzes and messages with embedded hyperlinks. When powered on, the localized verbal instructions lead users through the audio user interface. To access recordings, users are guided by audio prompts and respond with key presses.

However, even with high volume manufacturing, the production cost based on COTS hardware would likely exceed $20 per unit, which places any unsubsidized retail price out of reach for the target audience. Adding an additional integrated circuit to support radio reception and improve the scalability of content distribution would further add to this unit cost.

In addition, the operating cost depends upon the power draw of each of the components, including the sleep power. A pair of carbon-zinc D cells, typically found in rural markets for $0.35-0.40, provides 12-15 hours of power for typical use [14]. AA cells would allow the form factor to shrink to a pocket-size device, which, according to usability interviews, could increase usage and peer-to-peer copying. However, their substantially lower energy capacity reduces the device's useful lifetime far too much – to perhaps just 2-3 hours, optimistically.

It is for these reasons that we begin with the Talking Book and explore, in the rest of this paper, how to address these basic problems – *initial cost, network connectivity, and operating cost* – by extending the design space to include custom integrated circuits and leverage extant communications networks in the most economical manner.

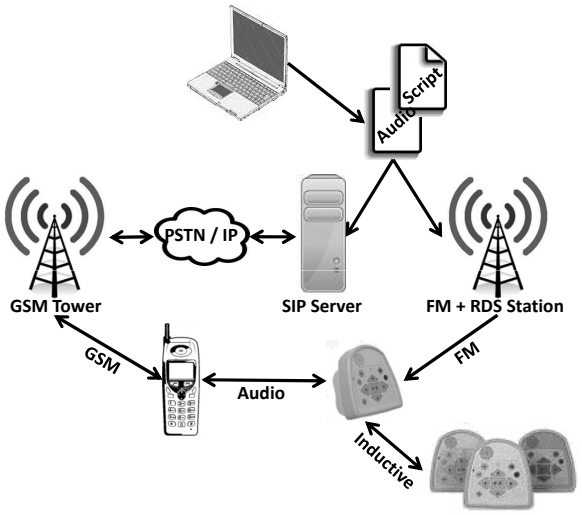


Figure 2: Rural audio computer system architecture and data flows for low-cost information delivery including over GSM voice/data, FM broadcast, and inductive peer-to-peer.

2.4 Implications for COTS vs Custom

The current COTS-based Talking Book includes a 16-bit, 96 MHz system on a chip with 256 KB on-chip ROM, a 1 MB NOR Flash chip for program code, and a 256 MB microSD card for audio storage. Accounting for 20% of the total bill of materials, the microSD card is the most expensive component. The voltage regulator, DC-DC converter, speaker amp, and more than 150 other components make up a relatively small portion of the cost but require PCB real estate accounting for 13% of the total cost. By bringing these components onto a single chip, not only can we reduce cost and supply risk, but we also allow for greater PCB flexibility.

3. SYSTEM ARCHITECTURE

Figure 2 shows the overall architecture of the rural audio computer system, including offline content creation, audio and script dissemination over FM broadcast (direct) and GSM voice (indirect, via mobile phone headset tethering), and local sharing using inductive links. The key computing and communications elements of the architecture include:

Host Computer. A host computer is used to create and package audio, metadata, and script content for distribution. These files are then delivered, through a variety of different channels, to the rural audio computer. The simplest channel involves directly connecting to an audio computer over the host computer's headphone port (not shown).

FM Broadcast. FM broadcast is the primary information delivery vehicle. Audio content is transmitted using the voice subchannel while metadata and scripts are transmitted using the worldwide radio data system (RDS) [13], if available. If not available, a radio station can be inexpensively upgraded to support RDS encoding and transmission.

GSM Network. Audio and data over GSM voice offers a secondary information delivery vehicle. A rural audio computer tethers to a mobile phone's headset port, and the mobile phone makes a call to a VoIP-terminated service. Audio and in-band data signaling are sent over this channel. The data are modulated using a GSM codec-friendly scheme.

Component Name	1.8V Power (mW)	3.2V Power (mW)
Cache	14.80	24.60
Amplifier	8.333	8.333
Radio	4.500	4.500
Cortex-M0	3.600	6.000
NAND Flash	3.600	3.600
Step-Up Converter	1.350	1.350
Clock Generator	0.625	1.041
Linear Regulator	0.067	0.111
Capacitive Sensor	0.002	0.003
Misc.	1.800	3.000
Total	**38.64**	**52.54**
Average	**45.59**	

Table 1: Active Power Draw. Note that efficiency drops at high battery voltage, resulting in higher power draw.

Rural Audio Computer. An audio computer can receive content from FM broadcast using its built-in FM receiver, over its headset port by tethering to a mobile phone or host computer, or in peer-to-peer mode using a built-in inductive link. For headset-based communications, a software modem is needed to transfer digital information. It is also able to record new messages directly for peer-to-peer dissemination or for feedback to government or NGOs.

4. DESIGNING A VIABLE PLATFORM

This section presents the architecture and design of a new generation of low-cost, communications-enabled *rural audio computers*. We highlight how the focus on a custom integrated circuit offers dramatic cost reductions – factors which will enable a $7.77 cost point in modest volume.

4.1 Processing

The two main considerations for the choice of processing core are energy consumption and performance. With approximately 7000 gates, from ARM's TSMC180 Standard Library, the ARM Cortex-M0 is an ideal choice. It can decompress Speex audio, constant pitch speed shift audio using Synchronized Overlap-Add Fixed Synthesis (SOLAFS), and manage system overhead simultaneously at 30 MHz. While the data path is 32 bits wide, the M0 uses thumb code executing 16-bit instructions as opposed to 32-bit. This allows for better code density and minimizes system memory size.

We developed an extensive power manager that allows the core to go into different low-power modes, and instructs certain peripherals to go to sleep if necessary. Analog blocks were designed to allow clock gating in order to reduce the overall sleep power to approximately 10 μA. During active mode, the core accounts for less than 10% of the total energy budget (compare Table 1).

The custom processor lets us remove unneeded components and incorporate only the bare essentials. For example, we do not require a separate program flash as code is loaded directly from NAND flash, saving the power and cost of a second (NOR) flash chip.

4.2 Memory Hierarchy

The memory hierarchy for the device is designed to meet three goals: (i) add as little cost to the system as possible; (ii) support a large code base as the current Talking Book application requires nearly 1 MB; and (iii) make it easy to use for programmers. To reduce cost we eliminate the NOR flash chip and store program code and data together with the file data in the NAND flash chip.

Component Name	Area (mm^2)	Percentage
Cache	21.000	77.20
Step-Up Converter	1.200	4.41
Cortex-M0	0.500	1.84
ADC	0.264	0.97
Capacitive Sensor	1.400	5.15
Linear Regulator	0.071	0.27
Amplifier	0.001	0.01
Misc.	2.764	10.15
Total	**27.2**	100.00

Table 2: IC area breakdown. The cache dominates chip area, as expected, and therefore dominates the chip unit cost. However, it obviates the need for either an external SRAM chip (for code/data) or an external NOR flash chip (for code).

Reducing the chip and pin count in this manner saves significant cost, but leads to a significant problem: the NAND flash chip is not fast enough to support instruction fetching without stalling. To address this problem, we use a 256 KB on-chip SRAM as a *cache* for program code. Structuring the SRAM as a cache frees the programmer from manually managing this resource (e.g., using overlays). The on-chip SRAM also stores volatile program data as the NAND flash would require several milliseconds to modify a variable. Section 5.3 describes how the software interacts with the cache.

Finally, the memory hierarchy includes a small (few KB) on-chip ROM, which stores the boot code necessary to initialize the peripherals, fetch the first instructions from the NAND flash, and reprogram the device if the external NAND flash happens to become irrecoverably corrupted.

4.3 User Interface

The *rural audio computer* employs an audio interface to circumvent the low literacy rates of its primary user population. However, external Class-A amplifiers are inefficient (50%). Instead, we designed an on-chip Class-D amplifier that increases the efficiency to 98%. Coupled with a carefully-chosen speaker, we can achieve audio volume levels considered normal during the pilot *Talking Book* program, at a power draw of 8 mW (Table 1), which marks a reduction of approximately 6×. In addition to the amplifier, an integrated 10-bit ADC with a 1-1000× pre-amplification stag and all the passive components to match the impedance of the microphone, further reducing component count and PCB size.

Many modern electronic devices use membrane-backed push buttons. This custom rubber membrane increases packaging cost by approximately $1 and adds mechanical failure points. We decided to add a Capacitive to Digital converter (CDC) to our chip, incurring an area penalty of less than 1% (see Table 2). The CDC allows us to replace the push buttons with a capacitive touch sensing array that determines the proximity of a finger outside of the case by measuring the change in capacitance. Figure 3 shows the CDC testboard with several different capacitive touch sensors on the PCB.

4.4 Communications

A significant improvement over the current *Talking Book* platform is the addition of wireless communication capabilities. For wide-area dissemination, we added an external FM radio receiver chip. A custom inductive link communication interface will support peer-to-peer communications, and connectivity over a mobile phone will be possible by transferring data through the audio headset port [9]. The following sections discuss each communications modality in detail.

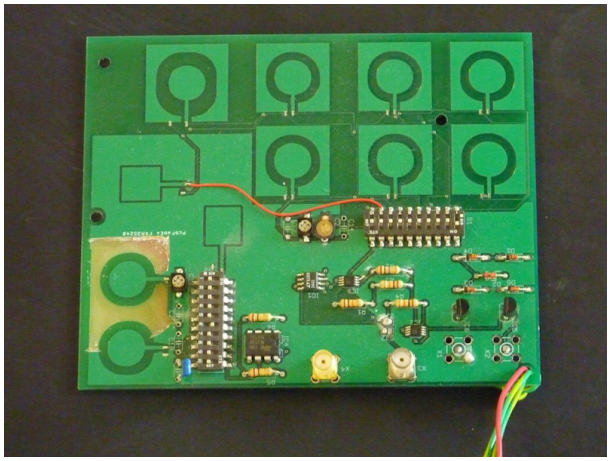


Figure 3: Capacitive digital converter (CDC) sensor testboard. CDC sensors can detect the change in capacitance resulting from the presence (or absence) of a finger. Therefore, it eliminates the need (and cost) of a custom elastomer membrane for detecting pushbutton presses. The remaining CDC signal processing functions can be placed inexpensively on silicon. A variety of capacitive touch pad geometries were prototyped to evaluate area, performance, reliability, and cost tradeoffs.

4.4.1 FM Radio

In order to receive wide-area programming, we use the Si4705 FM Radio receiver chip from Silicon Labs [16]. We chose an external chip rather than attempting to design and integrate the equivalent functionality onto our own silicon to mitigate numerous incompatibilities, risks, and challenges. Many parts of the radio chip are large analog components that tend to use older semiconductor process technologies that are incompatible with the rest of our custom silicon and would increase its size, cost, and power draw. Furthermore, while the cost of the Si4705 is significant, it adds substantial value to the platform as the chip can receive both FM audio and RDS concurrently. The only additional capability we will add to our own chip with regards to FM is a timer that allows us to schedule recordings at future times. That way, the user need not wait until a particular program is broadcast, and can instead have the content downloaded automatically.

4.4.2 Inductive Link

Our design supports lossless data transfer between a pair of devices by employing a custom inductive link. The advantage of using an inductive link to communication, rather than a regular wireless communications chip, is the low complexity of the transceiver. With a 5 cm×5 cm PCB coil trace we can achieve a bitrate of 500 kbps at a distance of up to 10 cm, while consuming just 50 nJ/bit and requiring no additional discrete components or integrated circuits.

4.4.3 Mobile Phone

To support communications with a remote server over the GSM network, we added a Universal Asynchronous Receiver/Transmitter (UART) controller to our chip. The UART controller is small (less than 1% of the chip area) and adds only two additional pins (RX and TX). The rest of the processing is done on the Cortex-M0. In addition, an audio crossover cable can be used to connect the device's earphone port to a mobile phone's headset port. Details about this interface are discussed in Section 6.2.

Component Name	Cost
Speaker Inductor	$0.0245
Speaker Capacitor	$0.0090
Linear Regulator Cap	$0.0107
Step-Up Converter Cap (Type 1) ×4	$0.0027
Step-Up Converter Cap (Type 2) ×2	$0.0246
Chip Area	$0.7600
Total Chip Cost	**$0.8315**
Chip	$0.8315
Speaker	$0.7783
Headphone Jack	$0.0232
Microphone	$0.0935
PCB	$1.0050
NAND Flash	$2.0000
Radio	$1.0395
Total Board Cost	**$5.7711**
Board	$5.7711
Casing	$1.0000
Test & Assemble	$1.0000
Total System Cost	**$7.7711**

Table 3: Bill of Materials. The system cost is dominated by the board cost which itself is dominated by the NAND flash and has roughly equal contributions from the custom chip, radio, speaker, and PCB. These components cannot be economically further integrated, suggesting addition gains from volume. Fractional cents come from the high volume pricing.

4.5 Power Supply

The power supply design is mostly driven by the availability of energy sources in developing regions. From extensive first-hand observations in Ghana, we found that Carbon Zinc batteries are the most common way of powering electronics (sans mobile phones). The reason is that access to grid electricity to recharge batteries is few and far between, solar cells are uncommon and expensive, and Carbon Zinc batteries are cheap and have long shelf life. The two most common form factors available are the D and AA sizes.

One D size Carbon Zinc battery holds almost 4× the charge of two AA batteries (8000 mAh vs. 2200 mAh) and costs approximately the same. However, the lower voltage of a single D battery requires that the processor and speaker amplifier operate from a boost converter, which incurs significant energy loss (∼60% efficient) and also adds cost from additional passive components. Further, the smaller form factor of the AA battery shrinks the overall system size, which reduces cost of the casing and, indirectly, the cost of shipping and distribution.

Carbon Zinc batteries are unique in that they have a very wide discharge voltage range, from 1.6 V to 0.8 V. In order to extract the full charge from two AA batteries, the input voltage range of 1.6 V to 3.2 V must be regulated to the 1.6 V minimum operating voltage of the cache. To accomplish this, we designed a low dropout voltage regulator which is able to regulate input voltages from 3.2 V to 1.95 V to the nominal 1.8 V supply voltage. Once the input voltage reaches 1.95 V, the linear regulator stops regulating and simply shorts the power supply to the battery and rides the battery voltage curve down to 1.6 V at which point a brown-out detection circuit asserts hard reset. This allows us to extract the maximum energy from the batteries, thereby reducing TCO.

While our custom silicon runs from the linear regulator, the external NAND flash chip and FM radio chip operate at 2.7 V. Therefore, we designed an integrated boost converter that uses a switched capacitor network to power these two chips. We chose a switched capacitor network instead of a buck converter due to the cost of the inductor (∼$0.38) needed for buck conversion. This inductor alone costs as much as the entire switch capacitor network (∼$0.27) while the efficiency is roughly the same. Furthermore, the boost converter cannot down regulate in case the battery voltage is at 3.2 V. The output voltage range of a switched capacitor network, however, is 1.8 V to 3.6 V. The efficiency is between 75%-85%.

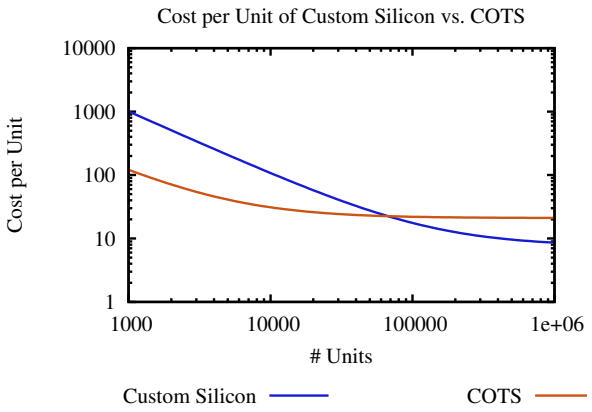


Figure 4: Unit cost vs volume for custom silicon and a COTS design. Custom silicon offers a lower price point than COTS above 70k units and achieves the target price point of $10 above 400k units while COTS cannot achieve a $20 price point at even 1M units. This analysis supports the case for custom silicon.

4.6 Platform Summary

The main goal of a custom chip design is to reduce cost. By integrating most components and their associated passives, we can significantly reduce part count, pin count, and power draw. This significantly reduces PCB design complexity, which further reduces size, manufacturing, and packaging costs.

Table 1 shows the power draw of the individual components. The size of our custom chip will be ∼27 mm^2 with 55 pads. Table 2 summarizes the size of each chip component. The cache takes up the majority of the area with 77%, followed by the step-up converter and capacitive sensor each with 5%. The rest consumes less than 13% of the total area, including the ARM Cortex-M0 core. Table 3 shows the bill of materials resulting from this aggressive integration. We expect that the next generation *rural audio computer* can be produced for less than the target cost of $10.

In Figure 4, we show the cost as a function of production volume. The cost of the custom silicon solution includes the initial mask cost needed to mass-produce the chip, which is approximately $240k in 0.18 μm technology. The relatively low mask set cost in an older technologies was a significant factor for choosing the 0.18 μm technology for this project. In addition, the design effort was estimated to be eight engineering years at $160k per year for a total engineering cost of $1.28M. In actuality, the engineering cost was significantly lower since the chip was designed by graduate students at the University of Michigan. Figure 4 shows that the cost of both the COTS and custom solution drops as volume increases due to the amortization of non-recurring engineering (NRE) costs and the reduced cost of discrete components at high volumes. However, the COTS solution remains above $20 even for 1M units while the custom solution reaches the required $10 price point at 400k units, motivating the need for a custom solution. The crossover point between COTS and custom solutions occurs at 70k units, which is relatively small compared to the size of the target market.

5. SYSTEM SOFTWARE

The usage scenarios and hardware design described above impact the design of the software running on the system. For example, the need to support programs with potentially large code sizes and the need to reduce the number of chips in the system led us to an SRAM-flash memory hierarchy. Implementing this memory hierarchy with a NAND flash chip required us to implement a flash translation layer in software, and this in turn required us to implement a software mechanism to handle memory accesses that encounter cache misses.

The software running on the rural audio computer has three main layers: application scripts, a script interpreter, and an operating system kernel. Application scripts are written in a simple markup language and describe a state machine. The script interpreter, which is written in C, parses and executes application scripts. The operating system kernel, which is written in C and ARM assembly language, provides standard operating system functionality, including boot code, interrupt handlers, device drivers, threads, and a FAT-32 file system. It also provides a flash translation layer and a cache miss handler, which we describe in more detail below. The script interpreter and kernel are compiled together to form the native ARM code running on the ARM Cortex-M0.

The rest of this section describes three notable components of the software in more detail. First, we describe application scripts, which make it easy for end users to write new applications for the rural audio computer. Second, we describe the flash translation layer, which provides robustness by guaranteeing atomic updates to logical sectors. Third, we describe how and why we handle cache misses in software.

5.1 App Programming and Script Interpreter

For the rural audio computer to address diverse needs in developing regions around the world, we need to ensure local software developers can design new applications, reconfigure or patch existing ones, and even patch lower-level system code. This also instills ownership in the project in participating countries and promotes long-term sustainability.

The kernel and script interpreter are primarily written in C. To expand developer support beyond C programmers, most device functionality is controlled by a declarative programming model executed by the script interpreter. Changing system menus, content navigation, and volume control are all possible by editing a text file. New audio applications can be built with the same programming model. Applications can include embedded hyperlinks from one segment of content to the next and can include conditional branching to allow for interactive and entertaining applications – a universally important driver of user adoption. Just as the early stages of the Web attracted thousands of new HTML programmers, this similar markup language opens the door to a broader base of device programmers.

The script shown in Figure 5 demonstrates a simple quiz with a true/false choice. In this example, each question and potential answer were recorded onto a separate Speex (.spx) file ('F'). Conditional branching is determined based on actions ('A') of the user (pressing a key, like '<' or '>') or the time points within a recording (e.g. the end of a recording, '$'). When a condition is satisfied, the action taken is often to jump to another recording using a human-readable label in brackets.

This example also shows that responses indicating whether the user selects the correct choice were recorded onto a single file, responses.spx. Separate audio prompts on a single file are distinguished by time blocks ('B'). The same file includes a block of time with instructions telling the user which buttons to press to listen to a new option (the left '<' and right '>' arrows) and how to pick an answer (the circle 'o'). These responses are inserted ('I') before proceeding to play the next audio recording.

```
F:q1.spx [question 1]
A$:I[instructions]
A>:[question 1-true]
A<:[question 1-false]

F:q1true.spx [question 1-true]
A<:[question 1]
A>:[question 1-false]
Ao:I[wrong] [question 1]

F:q1false.spx [question 1-false]
A<:[question 1-true]
A>:[question 1]
Ao:I[right] [question2]

F:responses.spx
B:0-750. [right]
B:1500-2250. [wrong]
B:4000-8500. [instructions]
```

Figure 5: An example application programming script.

To attract computer-savvy users who prefer to avoid declarative programming, an application with a graphical user interface is being developed to allow users to create and edit applications with drag-and-drop actions. This applications simply generates the markup code shown above, similar to a WYSIWYG HTML editor. Typical users of this application could include university students, employees of non-profit organizations, and district officers of the ministries of health, education, or agriculture.

5.2 Flash Translation Layer

Our rural audio computer uses a NAND flash chip to store audio files, most program code, and an initial data image (the rest of the program code and data is stored in boot ROM). NAND flash has several unique characteristics, such as the need to separately erase (setting bits to 1) and program (setting bits to 0) data, the need to erase a large block (e.g., 128 KB) of data, and a limited number of erases cycles on a single block before it wears out. The job of the flash translation layer (FTL) is to hide these limitations from higher system levels. Often, NAND flash is packaged together with an integrated FTL, such as in microSD cards. However, these packages incur a large overhead in power and access latencies, so we instead use a standalone flash chip and implement our own FTL that runs on the main ARM processor.

A major design goal for our FTL was robustness: we guarantee that each write of an individual sector (512 bytes) is atomic, even in the presence of power outages at arbitrary points in the device's operation (this guarantee is stronger than that provided by commercial microSD cards).

Our basic translation scheme maps logical blocks to physical blocks (a block is 128 KB). To allow programs to modify smaller units, we provide a data journal, which logs updates to 512-byte sectors [8]. When the data journal fills up, we merge the updated data for each block with the rest of the data in that block and store the new data in a free block. We store the current mapping from logical to physical blocks (and the locations of the data journal and any bad blocks) in a block-map journal, which contains a snapshot of the mapping plus any changes made after the snapshot was taken.

We use shadowing to atomically erase the block-map journal. We dedicate a block to store a one-bit pointer, which points to one of two fixed locations for the block-map journal. Using a one-bit pointer provides atomicity, since it can only transition between two states. To create a fresh snapshot of the block map, we first merge the old snapshot with the operations in the old block-map journal and write the new snapshot to the new block-map journal. We then toggle the one-bit pointer to point to the new block-map journal.

A sudden power outage can leave flash bits in an indeterminate state if it happens while writing or erasing, and this may cause the commit bits to return different values when read. To guarantee that inconsistent values are not read from commit bits, the kernel *stabilizes* the data and block-map journals during startup. It stabilizes the data journal by copying it to a new block and updating the block map, and it stabilizes the block-map journal by writing it to the other block-map journal and updating the one-bit pointer. The kernel stabilizes the one-bit pointer to the current block-map journal by re-erasing it to a 1 (if it is read as a 1 during boot), or re-programming to a 0 (if it is read as a 0 during boot).

Finally, we use block 0 of the flash chip to store the current location of the block-map journal and block-map journal pointer. Flash chips guarantee that block 0 will work correctly for the first 1000 erasures.

Other than block 0, the rest of the blocks on the flash chip are accessed through logical block numbers that are mapped through the FTL. This logical space is organized as a DOS partition table and two partitions: a boot partition that stores the kernel and script interpreter, and a FAT-32 file system partition that stores application scripts and audio files.

5.3 Cache and Memory Management

We use a single NAND flash chip as a consolidated storage device. NAND flash works well for file data, which is accessed in large units, but it performs poorly for program code and data as they are accessed a few bytes at a time. To improve performance, we use the on-chip SRAM as a cache for the program code and data stored on the NAND flash.

One novel feature of our caching strategy is how we handle cache misses. Instead of the traditional strategy of handling misses as part of the processor state machine, we deliver an interrupt to the processor and handle the cache miss in software (similar to what happens when accessing a non-resident page in a virtual memory system). We chose to handle cache misses in software for three reasons: compatibility with a software FTL, flexibility in prefetching data, and performance through non-blocking cache misses.

First, handling cache misses in software is needed to work with a software FTL. Program code and data are stored in logical blocks of the flash chip. Since these are accessed through the FTL, handling a cache miss requires calling the FTL code to read the missing data from the NAND flash.

To handle cache misses in software, we must guarantee that the code for the cache miss handler will always reside in the cache. To guarantee this, the cache allows the processor to define a pinned address range: data within that address range will not be evicted from the cache once filled. During the initial boot stage, the boot ROM loads the cache miss handler and FTL into the pinned cache region, thereby ensuring that they will be present in the cache for all future cache misses. We store all writable data in the pinned region to minimize expensive writes to the NAND flash.

The second reason we handle cache misses in software is to gain flexibility in prefetching data. The software is in complete control of its internal buffering, allowing it to prefetch any amount of the cache on each cache miss. Currently, we prefetch an entire flash page (2 KB) on each miss into a software buffer. The software cache miss handler could also perform more complicated optimizations, such as profiling the workload and prefetching data accordingly. The software can also move critical code into the pinned region or adjust the size of the pinned region.

The final reason we chose to handle cache misses in software is to improve performance through non-blocking cache misses. If a cache miss occurs and the required data are not present in the software buffer, the cache miss handler can switch to another thread. By doing so, the operating system can provide real-time guarantees for playing audio, even if non-real-time code misses in the cache.

The operating system separates cache handling into a top half and bottom half handler. When a cache miss occurs, the processor receives an interrupt and begins executing the top half cache miss handler. In the common case, the missing line is in the software buffer, and the top half handler will copy that data out of the buffer into the cache and return. If the data is not in the software buffer, the top half handler sends a request to the bottom half handler to load the data from the NAND flash. After making this request, the top half suspends the thread that invoked the cache miss and switches to the next runnable thread. The top half loads the requested data through the FTL, updates the software buffer, and marks all threads waiting on the cache line as runnable.

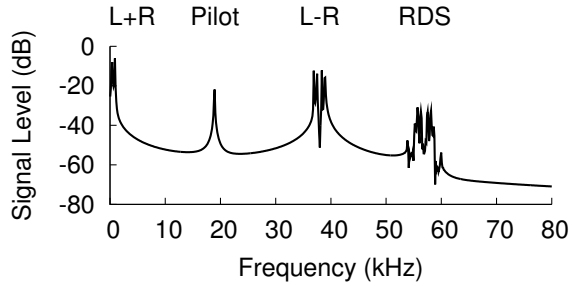
6. CONTENT DISTRIBUTION NETWORK

The content distribution network is responsible for disseminating audio, metadata, scripts, and firmware to geographically distributed rural audio computers, and enabling peer-to-peer communications between the books for content sharing and firmware upgrades. The key challenge lies in transferring these data at low cost in places with often poor connectivity. We plan to address this problem by preferentially transferring audio, metadata, and scripts principally over FM broadcast and, in (hopefully) infrequent cases, over the GSM voice channel. Although the former method is well-known to be viable, the latter method raises several challenges and is not well-explored in the research literature. We also support direct tethering, over the headset port, to music players, mobile phones, and laptop computers, as well as inductive communications for peer-to-peer data transfer.

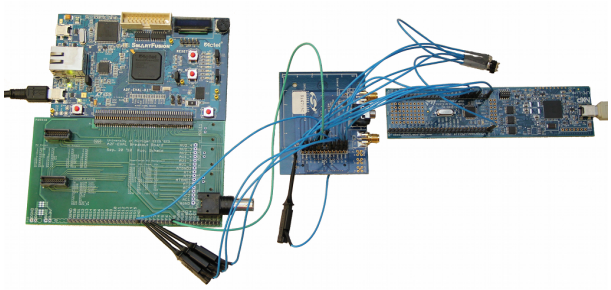
6.1 Broadcasting Content over FM

Distributing content using FM broadcast allows us to leverage existing technology and an asymmetric cost structure that favors rural audio computers. Content created using a conventional computer is packaged and sent to a radio station. The package includes audio, metadata, and script files, along with a content index and broadcast schedule. An FM station broadcasts the index digitally using FM's worldwide RDS data channel [13] at pre-arranged, periodic intervals. If a station does not have RDS capability, it can be retrofitted inexpensively, since commercial-grade RDS encoders cost between $900 and $2900, depending on supported features [5]. GNURadio offer a low-cost prototyping alternative, which we use.

A rural audio computer receives the index, perhaps daily, and then its owner can select which specific programs to download based on the index. The index includes schedule information about the programs to be transmitted subsequently and is itself transmitted as both an audio program and its digital representation. Since the index can be small, and is encoded digitally, its transfer over RDS is fast and efficient. Once specific audio programs are selected from the index, the audio computer will tune in at specific times to download the audio content of interest. The audio and metadata are transmitted on the FM band simultaneously, at the scheduled time, and stored locally on the audio computer.



(a) FM/RDS baseband spectrum



(b) I2S on an FPGA (c) FM receiver (d) Microcontroller

(e) FM broadcast (f) Received audio

Figure 6: FM/RDS (a) baseband spectrum and (b)-(f) prototype receiver subsystem and operation. A GNURadio device operating as a combined FM/RDS station (a) transmits combined FM audio and RDS digital data to (c) an FM/RDS radio receiver. The receiver is controlled via its SPI interface by (d) an ARM Cortex-M0 microcontroller that approximates the processing core in our custom chip. The FM/RDS receiver demodulates (e) the FM carrier and (f) outputs a digital audio stream (I2S protocol) and RDS data (I2C or SPI), to (b) a custom Verilog peripheral that implements an AHB bus bridge and runs on an FPGA. All of the electronics shown here, except for the radio, will be integrated into our custom chip.

Using this two-step pull process ensures that energy is consumed principally downloading content that is of interest. In addition, while the index is very short, perhaps just a few tens of seconds, many programs are substantially longer, spanning many minutes. Therefore, to minimize flash usage, audio data are digitized and compressed. The digitization can be performed either by the audio computer's processor, or using the SiLabs Si4705's [16] digital audio output option. The latter choice eliminates an unnecessary DAC/ADC conversion step, and its associated power, but the majority of reception power goes to the radio receiver, processor, and flash memory, so minimizing the amount of unrequested content dramatically improves battery life which reduces TCO.

Figure 6 shows the FM/RDS spectrum and our receiver prototype. The FM pilot tone, mono (L+R), stereo (L-R), and RDS subbands are clearly visible. The prototype hardware includes the radio receiver, a microcontroller, and a custom Verilog I2C interface. All of these electronics will be integrated into just two chips.

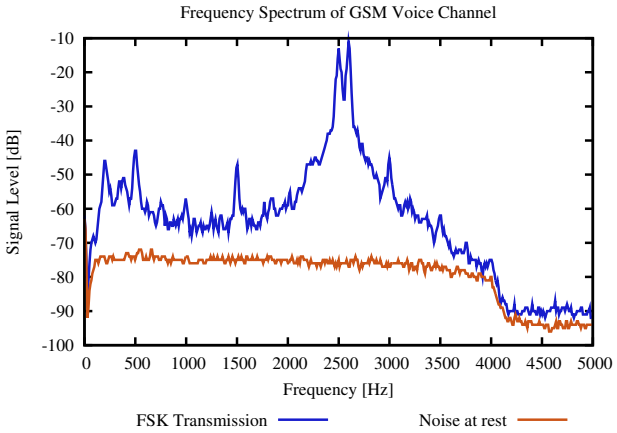


Figure 7: Frequency spectrum of the GSM voice channel at rest, and during a modulated transmission. Our goal is to transmit metadata concurrently with audio. By shifting the center frequency of the FSK communication close to the upper spectrum limit near 3500 Hz, we leave a band from 300 Hz to 2500 Hz available for audio.

6.2 Unicasting Content over GSM Voice

In some cases, broadcast FM is not suitable for dissemination, including when a local FM station is not available or when the content is large and primarily digital (e.g. a firmware upgrade) or if feedback to the source is needed. In such cases, we leverage existing GSM voice networks by transmitting either data alone, or audio and data over voice. As in the case of FM broadcast, audio, metadata, scripts, and firmware are packaged and placed on a server. Then, an audio computer is connected to a mobile phone over the headset port. Next, the phone is used to dial a local number which is connected, via a PSTN-to-VoIP gateway, to a server that terminates the voice call. The remote server and audio computer exchange audio and data over the GSM voice channel.

Prior work has shown how data rates of up to 4 kbps are possible over the GSM voice channel [10]. While the encoding and modulation of the particular scheme is computationally low, decoding is another matter, and is quite computationally intensive, and thus not amenable to the audio computer platform using traditional decoding techniques. Additionally, the method proposed in the literature does not constrain the symbols to a specific frequency range, and thus digitally removing them from the voice stream is difficult (in the case of concurrent voice and data). Of course, serializing audio and data over the GSM voice channel may be another design option worth exploring.

Since our bandwidth requirements are relatively low, and simplicity of the modulation scheme is important to lower cost, we currently use binary FSK modulation with center frequency at 2.5 kHz and a frequency separation of 100 Hz. Our initial prototype achieved 50 bps at a byte error rate of 12.4%, suggesting a greater than 1% BER. Figure 7 shows the GSM voice channel frequency spectrum at rest, and during a modulated transmission. The two peaks show the mark and space symbols, and we can see how the channel quickly drops off after 3500 Hz. To avoid interfering with analog voice audio, we plan to move the center frequency of our modulation to 3400 Hz, and digitally filter out the voice and data components of the transmission. Improving the demodulator design will also increase the data rate.

We have also explored overlaying digital FSK communication over regular GSM voice communication, and have found that the main challenge to overcome is the non-linear, non-memoryless GSM voice channel itself. In contrast to more common communication channels which are memoryless, the GSM voice codec maintains state about recent transmissions so audio signals that do not exhibit voice-like characteristics are distorted by the encoding and decoding processes of the codec. How the GSM channel distorts conventional modulation schemes has been described in the recent literature [6].

6.3 Disseminating Content Peer-to-Peer

A number of usage scenarios are best served with peer-to-peer data exchanges. These include sharing individually recorded messages, downloaded audio content, usage statistics, and firmware upgrades. Peer-to-peer communications that mirror human inter-contact networks, and allow viral data exchange, are useful because inexpensive and scalable mechanisms for bi-directional communications are rare (recall that in most cases, mobile phones are too expensive). We plan to support such communications using triggered exchanges. That is, when two rural audio computers are placed next to each other and triggered (e.g. by a button press), they exchange the selected content, as well as any system-queued data, like firmware updates. The Talking Book does this today over USB; our next-generation rural audio computer will reduce cost and increase reliability through inductive link-based communications.

7. CONCLUSION

Affordable access to on-demand, personally-relevant information has transformed lives in developed nations. We routinely take the Internet for granted to research potential illnesses, choose schools for our children, and determine the best time to plant daffodils. Unfortunately, extreme cost sensitivity, limited power and network access, and widespread illiteracy place even the most basic and critical information about health, agriculture, and education out of the reach of rural villagers in developing nations, even though studies suggest that access to such information has statistically significant and measurably positive impact.

To help bridge this growing digital divide, we present the design and preliminary results of a new rural audio computer system for low-cost information access targeted to the particular constraints of developing regions. Our work highlights the key role that custom integrated circuits play in driving costs down, the importance of active and standby power on system design, the options for hardware/software/network co-design tradeoffs, and the challenges with creating a viable content distribution network. With such large market volumes at stake, the seemingly high NRE costs of custom silicon are far outweighed by the tremendous upside cost advantages at these same volumes. In some cases, like the one we present here, *only custom silicon can make the solution economically viable for individual ownership.*

8. ACKNOWLEDGMENTS

Special thanks to Aaron Schulman for prototyping the FM/RDS receiver subsystem, Mark Brehob for discussions about data communications over GSM voice, and Eric Brewer for originally suggesting we investigate inductive coupling for low-cost, peer-to-peer communications. This material is supported in part by National Science Foundation Award #0964120 ("CNS-NeTS"). Additional NSF support was provided under Grant #1019343 to the Computing Research Association for the CIFellows Project. Literacy Bridge's work is supported in part by a grant from Amazon.com.

9. REFERENCES

[1] http://www.itu.int/ITU-D/ict/statistics/material/Africa_Village_ICT_2007.pdf.

[2] Asterisk: The open source telephony project. http://www.asterisk.org/.

[3] MobileActive. http://mobileactive.org/countries.

[4] SMS Cost in Africa. http://manypossibilities.net/2009/04/sms-costs-in-africa-2008/.

[5] Audemat. RDS Encoders: FMB80/FMB10/RDS Silver. http://www.audemat-aztec.com/ressources/brochure_download/rds%20encoder%20range_bro_en_v1.0.pdf, 2010.

[6] A. Dhananjay, A. Sharma, M. Paik, J. Chen, T. K. Kuppusamy, J. Li, and L. Subramanian. Hermes: data transmission over unknown voice channels. In *MobiCom '10: Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 113–124, 2010.

[7] R. Gandhi, R. Veeraraghavan, K. Toyama, and V. Ramprasad. Digital green: Participatory video for agricultural extension. *Information Technologies for International Development*, 5(1):1–15, 2009.

[8] J. Kim, J. M. Kim, S. Noh, S. L. Min, and Y. Cho. A space-efficient flash translation layer for CompactFlash systems. *IEEE Transactions on Consumer Electronics*, 48(2):366–375, May 2002.

[9] Y.-S. Kuo, S. Verma, T. Schmid, and P. Dutta. Hijacking power and bandwidth from the mobile phone's audio interface. In *DEV'10: Proceedings of the First Annual Symposium on Computing for Development*, Dec. 2010.

[10] C. LaDue, V. Sapozhnykov, and K. Fienberg. A data modem for GSM voice channel. *IEEE Transactions on Vehicular Technology*, 57(4):2205–2218, 2008.

[11] G. Legros, I. Havet, N. Bruce, and S. Bonjour. *The Energy Access Situation in Developing Countries: A Review Focusing on the Least Developed Countries and Sub-Saharan Africa*. UNDP & WHO, New York, 2009.

[12] N. Patel, D. Chittamuru, A. Jain, P. Dave, and T. S. Parikh. Avaaj otalo - a field study of an interactive voice forum for small farmers in rural india. In *CHI'10: Proceedings of ACM Conference on Human Factors in Computing Systems*, 2010.

[13] RDS Forum. The new RDS IEC 62106:1999 standard. http://www.rds.org.uk/rds98/pdf/IEC%2062106-E_no%20print.pdf, 1999.

[14] C. Schmidt, T. J. Gorman, M. S. Gary, and A. A. Bayor. Impact of low-cost, on-demand information access in a remote ghanaian village. In *ICTD'10: Proceedings of International Conference on Information and Communication Technology and Development*, Dec. 2010.

[15] J. Sherwani and et al. Orality-grounded hcid: Understanding the oral user. *Information Technologies and International Development*, 5:37–49, 2009.

[16] Silicon Laboratories. Broadcast FM radio tuner for consumer electronics. http://www.silabs.com/pages/DownloadDoc.aspx?FILEURL=Support%20Documents/TechnicalDocs/Si4704-05-C40.pdf, Dec. 2009.

[17] United Nations Education, Scientific and Cultural Organization. Education for all global monitoring report 2010, reaching the marginalized. UNESCO, Paris, France, 2009.