A Dual-Processor Solution for the MAC Layer of a Software Defined Radio Terminal

Hyunseok Lee, Trevor Mudge
Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
{leehzz,tnm}@umich.edu

ABSTRACT

Considerable work has been devoted to studying flexible computation structures for the physical layer of a software defined radio (SDR) terminal. However there has been almost no research on protocol processors for the upper layer protocols such as the media access control (MAC) and link protocol. A general purpose processor (GPP) is sufficient for the protocol processing of a single mode terminal. However, in the case of a multi-mode system required for SDR, there is a very wide set of possibilities for the MAC layer. In principle these too could be handled by a GPP. However, we show that a better solution is to use a GPP augmented by a small supplemental processor. The GPP is responsible for the relatively complex protocol operations in the active mode, and the supplemental processor handles the idle mode operations. This separation of responsibilities simplifies the implementation of hard real-time responses required by some protocols (for example IEEE 802.11), while maintaining the programmability needed to handle a wide range of protocols. In addition, this organization allows a significant power savings in the idle mode. This is important because the protocol processor must process a large number of tiny idle mode tasks whose aggregate effect over time dominates the power consumption in a wireless terminal.

As part of our study we develop a hardware model of the supplemental processor in Verilog and its software model in C. Using commercial CAD tools we synthesized out design and evaluated the power consumption and response time of the platform. Our results show that the proposed platform meets the real-time deadlines at low power while maintaining programmability.

Categories and Subject Descriptors

C.1.4 [**Processor Architectures**]: Parallel Architectures–Mobile processors

General Terms

Design, Performance

Keywords

Wireless platform, SDR terminal, Protocol processing

1. INTRODUCTION

Software defined radio (SDR) is a wireless communication system whose function blocks are implemented by software routines so that various wireless protocols can easily be supported by simple software changes. The concept of SDR originated in the military, but now it is emerging as important commercial technology. For instance, 4G, the next generation of wireless communication systems requires that multiple wireless protocols be supported [15]. Although the concept of SDR is very attractive, there are many obstacles on the way to its commercialization; the particularly power consumption.

There has been a lot of research effort on the physical layer but there has been almost no research on protocol processors executing upper layer protocols. In the past this was implemented on a flexible general purpose processor (GPP) for single mode terminals. In this paper, we analyze the impact of multiple wireless protocols on the architecture of the protocol processor and propose a dual-processor hardware solution.

To derive the requirements of the protocol processor we analyzed a collection of differnt wireless communication standards including Bluetooth, IEEE 802.11a/b/g, GSM, IS-95, IS-2000, and W-CDMA. One important requirement is the need for a tight response time of 10 us in the case of the IEEE 802.11 protocol. Fast response time is needed because the whole channel is occupied by a terminal when it is generating a response message. Thus, the protocol processor must respond quickly in order to maximize channel throughput. In addition, as explained in Section 3.2, wireless terminals perform many operations even during idle periods. A common characteristic of these idle period operations is a light workload - a simple field matching in most cases. Although the power consumption of each idle operation is very small, the aggregated idle power becomes dominant because a wireless terminal is idle most of the time.

As shown in Figure 1, we propose a heterogeneous dual-processor architecture consisting of a conventional main processor and a simple supplemental processor. The main pro-

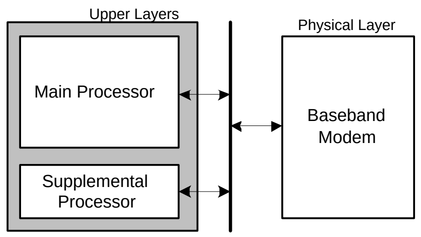

Figure 1: Proposed hardware platform for upper layer protocol processing in an SDR terminal

cessor provides the processing power needed for heavily loaded active state protocol operations, whereas the supplemental processor provides an adequate but power efficient processing capability for the idle mode. As we will show in Section 4.1, other candidate platforms employing a single GPP or a GPP plus ASICs have problems with scheduling or programmability.

The reason for using a GPP with minimal functionality as the supplemental processor is to achieve programming flexibility. However a GPP is power inefficient and slower than an ASIC solution. To see how these points impact the system we develop both hardware and software models for the supplemental processor and evaluate it with commercial CAD tools. The evaluation results are presented in Section 5. They show that the supplemental processor provides flexibility with very little power overhead.

The use of multiprocessor architectures for power reduction purposes is not new. The paper by Olsen showed that a heterogeneous multiprocessor architecture is power efficient for hand held devices such as a personal data assistant (PDA), because it allows the user idle period to be efficiently managed [14]. The paper by Kumar showed that a heterogeneous multiprocessor with a single instruction set architecture (ISA) is also power efficient even for general applications [9]. The contribution of our study is to apply the heterogeneous multiprocessor concept to SDR for the upper layer protocol processing. There have been several studies on small microprocessors that consume less than 1 mW power employing asynchronous logic or sub-threshold circuits [7][12]. However, in this work we implement the supplemental processor using a conventional design flow, because results in Section 5.5 show that any additional power gain will not be significant.

2. OVERVIEW

2.1 Wireless Communication Networks

Roughly speaking it is possible to classify wireless communication systems into three types: wireless personal area networks (WPAN), wireless local area networks (WLAN), and wireless wide area networks (WWAN). The WPAN system enables users to connect to various personal devices over short distances without wires. An application might be synchronizing data between a PDA and a desktop computer. Bluetooth is the most popular system at the moment.

WLAN systems originated from wired LAN systems. They aim to replace existing wired LANs by high speed wireless channels. A typical terminal in this network is a laptop having wireless access. IEEE 802.11a/b/g systems are the most widely used WLAN systems.

Finally WWAN systems evolved from a telephone network. The early generation of WWAN systems such as AMPS, GSM, and IS-95 provide voice service without spatial limitations. The 3rd generation systems provide multimedia services like video telephony on wireless channels. The CDMA-2000 and W-CDMA systems are common example.

In the future, we will need terminals that simultaneously support many of the above protocols to seamless span WPANs, WLANs, WWANs, plus their future derivatives.

2.2 Wireless Protocol Stack

We can divide the wireless protocol stacks into two categories by workload characteristics: the physical layer and the upper layers. The physical layer consists of various realtime and computation intensive operations such as forward error correction and modulation. Due to the tight power budget and high computation requirements it is common to implement the physical layer in ASICs.

In contrast the upper layer protocols consist of various control intensive operations such as media access control, retransmission of corrupted frames, terminal mobility support, and radio resource control. Thus the upper layer protocols are implemented on a GPP except for some hard realtime operations such as MAC response generation and encryption/decryption.

Although both physical and upper layers are within the scope of SDR, we only focus on upper layer protocol processing in this paper.

2.3 SDR terminal

Converging the features of several single mode terminals into one SDR terminal allows us more convenient wireless network access. For instance, we commonly use both cellular phone and wireless LAN card. A cellular phone provides a low speed but seamless wireless connection whereas a wireless LAN card provides a high speed Internet connection at limited area. The SDR technology enables us to adaptively select optimal network according to environment.

Furthermore there exist several different wireless protocols even for almost identical services, for example W-CDMA and cdma2000. Some countries deploy the W-CDMA system for wireless multimedia service and other countries do cdma2000. The SDR allows international roaming with one terminal.

3. SYSTEM REQUIREMENTS

3.1 Hard Realtime Response

The IEEE 802.11 network differs from other wireless networks in having short intervals between control packets as part of the data transmission. As shown in Figure 2, the transmitter and receiver exchange the request to send (RTS) and clear to send (CTS) MAC control packets prior to data transmission. This reserves a wireless channel, and the receiver transmits an acknowledge (ACK) packet after data transmission to release the wireless channel. Because the reserved link cannot be used by other terminals while preparing a MAC response, the IEEE 802.11 protocol requires a very fast MAC response in order to maximize network throughput. For the MAC response generation, we need to

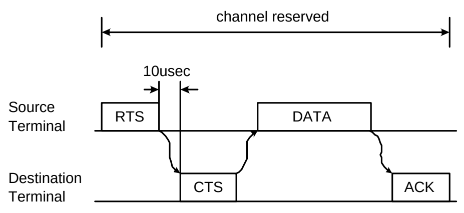

Figure 2: Data transmission procedure for IEEE 802.11 networks

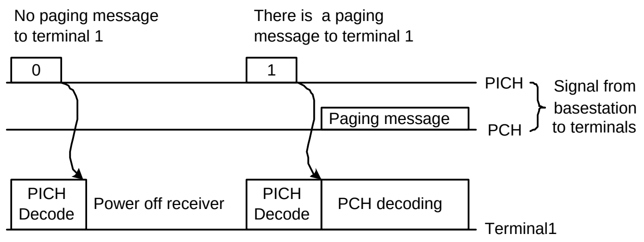
No paging message to terminal 1
There is a paging message to terminal 1
0
1
PICH
PCH
Paging message
Signal from basestation to terminals
PICH Decode
Power off receiver
PICH Decode
PCH decoding
Terminal1

Figure 3: Paging Procedure of cdma2000 and W-CDMA networks

analyze the header of the received MAC packet after decryption.

3.2 Processing Tiny Operations

3.2.1 Paging in cdma200 and W-CDMA

Paging is a procedure whereby a network identifies the base-stations having the best wireless paths to a destination terminal before establishing a user session. A terminal must be in its ready state to receive paging messages from base-stations during idle mode. WWAN networks have a two phase paging procedure so as to minimize the power consumed for paging message reception. In phase one, terminals periodically wake-up and decode the paging indication channel (PICH) that carries simple on/off information indicating the existence of a paging message in the following paging channel (PCH) slot. The bits on the PICH are not protected by a forward error correction scheme such as a convolutional code. The reason for using an uncoded channel is to save power by powering off error correction, because it is one of the most power consuming blocks in a wireless terminal. In phase two, the terminals that have detected the transmission of a paging message through decoding the PICH start decoding the PCH with full functionality including error correction.

In Figure 3, the first PICH bits indicate that there is no paging message in the following PCH slot, so the terminal turns of its receiver to save power. However the second PICH indicates that a paging message will be transmitted in the following PCH slot. Thus the terminal starts PCH decoding. Although PCH decoding is a small task, the protocol processor must be activated, because the paging message includes upper layer information.

3.2.2 Location Update in cdma2000 and W-CDMA

During the idle mode, the terminals of a WWAN can drift in any direction. If there is no rough estimation of a terminal location, a network needs to broadcast a paging message on the entire network. This is inefficient and avoided by a location update procedure. Several base stations logically comprise a location update area. Whenever a terminal passes the boundary of a location update area, it sends a location update message to the base-stations in order to notify that it is moving to an adjacent location update area. Thus the network can limit the broadcasting range of a paging message to within the current location update area.

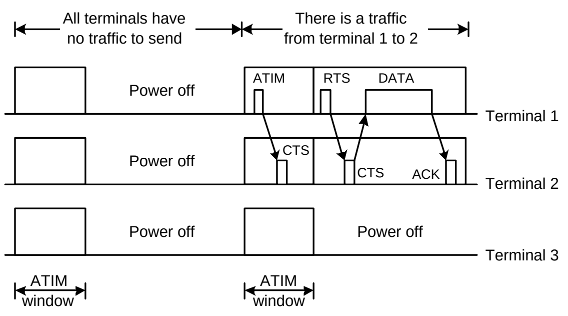

Figure 4: Power management procedure of IEEE 802.11 network

3.2.3 Power Management Procedures in IEEE 802.11 Network

In an IEEE 802.11 network, packet transmission is only allowed during the ad hoc network indication map (ATIM) windows. This restriction is to minimize terminal power consumption during the idle periods between packet bursts. As show in Figure 4, all terminals simultaneously wake up in ATIM windows. The terminals that have buffered traffic exchange ATIM request and ACK messages with the destination terminals, so that the destination terminals remain awake in the following idle period. In the interval between ATIM windows, source and destination terminals perform actual traffic transmission while the remaining terminals turn off their receivers to save power. To decode the ATIM packet, the MAC layer and the encryption engine must be involved.

3.2.4 Operation States

WWAN and WPAN networks have several operation states for terminal power saving and better radio resource utilization. Busy periods, short idle periods, and long idle periods between packet arrivals are mapped on different operation states that have different power and radio resource overhead requirements. A W-CDMA network has Cell_DCH, Cell_FACH, Cell_PCH, and URA_PCH states, and a CDMA-2000 network has Active, Control Hold, Suspended and Dormant states. Similarly Bluetooth has Active, Sniff, Hold and Park states. Among the above states, Cell_PCH, URA_PCH, Suspended, Dormant, Sniff and Park states are defined for idle period handling. They should be low power state. In these states the terminal performs various operations based on simple field matching.

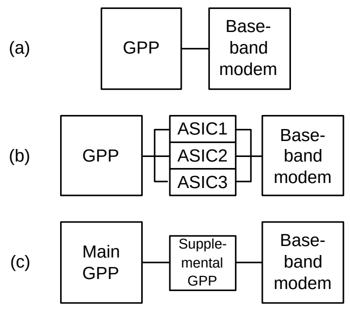

Figure 5: Candidate platforms for the protocol processor of an SDR terminal

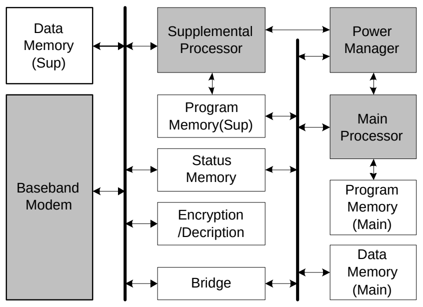

Figure 6: Proposed dual-processor platform for the protocol processor of an SDR terminal

4. PLATFORM DESIGN

4.1 Platform Comparison

For multiple wireless protocol processing we can consider three types of processor platforms: 1) a single GPP; 2) a GPP with ASICs; and 3) two or more GPP cores. The single GPP based platform, shown in Figure 5(a), provides the highest level of flexibility but has a scheduling problem. As we saw in Section 3.1, the protocol processor must generate a MAC response message within 10 usec in order to support the IEEE 802.11 protocol. In addition generally it is necessary to use a realtime operating system (RTOS) to cope with the algorithm complexity of the upper layer protocols. However, RTOSs are not able to context switch that fast e.g. uClinux, a Linux based RTOS, requires more than 10 usec [5]. This rules out organization (a).

Another candidate platform is shown in Figure 5(b). A GPP is assisted by ASICs that cover idle periods and realtime operations. Many commercial IEEE 802.11 modems are based on this architecture to satisfy hard realtime response requirement [11][4][8]. The GPP can be scheduled efficiently because the ASICs take over the burden of hard realtime MAC protocol processing, and ASICs are also power efficient. However, it is difficult to accommodate future protocol extensions with this architecture, because of the hardwired nature of ASICs.

As an alternative to either of these extremes, we propose the dual-processor platform shown in Figure 5(c). Because both processors are programmable, they provide a high degree of flexibility. Two processors can be schedule independently by a task partitioning: complex tasks requiring an RTOS on the main processor, and simple hard realtime tasks on the supplemental processor. The simple tasks on the supplemental processor can be programmed without an RTOS. It allows us to meet the requirement of hard realtime response. As we will see in Section 5.5, the power consumption of the supplemental processor is almost as low as an ASIC without sacrificing flexibility.

4.2 Task Partitioning

In this subsection we explore the task partitioning and its mapping to the hardware. Two major issues of the task partitioning are the number of tasks mapped onto the supplemental processor and the complexity of the interface between two processors. In this paper we propose assigning only idle period tasks and hard realtime tasks to the supplemental processor. This assignment minimizes power and simplifies the interface between the two processors. The number of tasks that can be placed onto the supplemental processor is limited by the hard realtime scheduling requirements. If complex algorithms are mapped onto the supplemental processor, it becomes difficult to program without the assistance of an RTOS. As we discussed in Section 4.1, it is impossible to meet the response time requirements if we use an RTOS.

In addition, the heavy workload would result in complex hardware and raise power overhead during idle period operations. The interface between two processors remains simple only if we map idle mode tasks onto the supplemental processor. Most idle period tasks discussed in the previous section are essentially just a simple field match except for the IEEE 802.11 networks where additional decryption of packets may be required. Therefore the length of the data needed for this field matching is less than packet header size - smaller than several tens of bytes in wireless communications. The supplemental processor's program can thus be very simple and does not require the support of an RTOS.

4.3 Proposed Platform

4.3.1 Main Processor

The data rate of applications executed on the main processor varies widely from voice at 9.6Kbps to packet data at several Mbps. To support such widely differing date rates with minimum power consumption, dynamic voltage scaling (DVS) and dynamic frequency scaling (DFS) techniques are essential for the main processor. The quality of service (QoS) parameters, determined at an initial session setup, can be used to indicate voltage and frequency scaling. For example, information that an established session has a constant bit rate permits aggressive processor scheduling without any slack. At short idle periods between packet bursts, a clock gating technique can be applied instead of DVS or DFS. Because the supplemental processor covers long idle periods the main processor has no need for leakage current reduction schemes such as adaptive body biasing - it can simply be shut off. This simplifies the main processor if it has a supplemental processor compared to the typical low power GPP for other portable devices [13][6].

4.3.2 Supplemental Processor

The simplicity of tasks performed on the supplemental processor allows us to minimize its functionality. The instruction set of this processor consists only of the essential instructions required to perform small control actions like paging message decoding. The width of the datapath needs to be 32 bit for fast packet processing. However, we used a processor model with an 8 bit datapath for our experiments because it was difficult to find a small 32 bit commercial model with a complete programming environment. In Section 5.3.2, we will show how a 32 bit datapath yields better power performance than an 8 bit datapath. Thus our power studies using a 8 bit processor are a worst case.

The supplemental processor requires a timer to generate events because many operations in wireless protocols are timer based. The timer events are processed by interrupt handling logic. The supplemental processor has two interrupt ports, one for an interrupt from the baseband modem indicating the arrival of a new packet, and another port for an interrupt from the timer. A clock gating technique is used in the supplemental processor to reduce the power dissipation during idle periods between interrupt events. The timer, interrupt handling logic, and clock gating provide an efficient implementation method for event driven applications.

In addition, the supplemental processor is equipped with a hardware comparator. The hardware comparator is designed for the field matching operation that the supplemental processor is frequently called on to perform.

The supplemental processor needs a hardware random number generator (RNG) that is used for a random back-off operation performed in the MAC layer. The execution time of a software based RNG is too long to meet the timing requirements. The randomness required for the back-off operation need not be of a high quality, so a linear feedback shift register (LFSR) is adequate. However we need to consider the impact of the initial seed of the LFSR. The initial value of a flip-flop is not random because small differences in the size of the transistors in a flip-flop lead to bias. This in turn can induce a load sharing problem in the ad hoc mode of the IEEE 802.11 network. The overhead of periodic beacon packet transmission is unevenly distributed if we use a LFSR without a proper random seed. To avoid such a situation we place a direct access path so that the power manager can load the initial seed into the LFSR.

4.3.3 Memories

As shown in Figure 6, the memory in our platform can be classified into three types, program memory, state memory, and data memory. The program memory stores the execution code of the processors. The program memory of the supplemental processor is accessed by both the main and supplemental processor. The main processor loads the execution program of the supplemental processor prior to activating it. A single port SRAM is used for the program memories of both processors.

Meanwhile the state memory stores protocol state information such as a terminal identifier and the QoS parameters of established sessions. When supporting the IEEE 802.11 protocol, both the main and supplemental processor need to be activated at the same time and simultaneously access the state memory. Thus a dual port memory is used for the state memory. The size of this memory varies according to the protocol. However, because the supplemental processor needs just a small portion of the state information, the size of the state memory need not exceed several hundred bytes.

The data memory stores user traffic processed by the protocol layers. The baseband modem stores packets which are received from antenna at the data memory of the supplemental processor. The processed packet is passed to the main processor by memory copy to the data memory of the main processor. For transmission, packets are copied in the reverse direction.

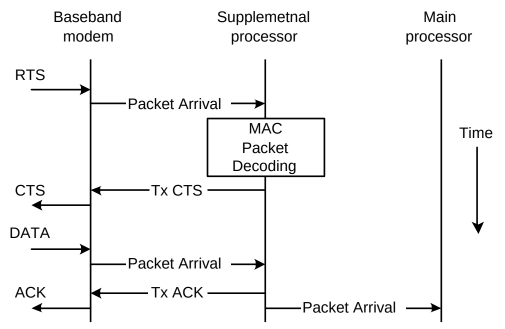

Figure 7: Packet reception procedure while executing the IEEE 802.11 protocol

4.3.4 Power Manager

A power manager controls the power state of all blocks in a wireless terminal. Therefore, it is always powered even at idle periods. It has a timer to trigger a periodic wake-up event. There are three input ports in the power manager. One input port from the main processor is used to configure the operation mode of the power manager according to the operating state change of a protocol. Another input port is assigned to the supplemental processor. Through this port, a request for the activation of the main processor is transmitted when the supplemental processor receives a communication request. The third port is connected to the baseband modem. If the baseband modem detects a signal on a PICH, it requests activation of the supplemental processor for PCH decoding through the third interrupt port.

4.4 Application Examples

Because the protocol processor discussed in this paper aims to support multiple wireless protocols, its operation procedure changes according to the type of protocol loaded onto the platform. Figure 7 shows the interaction between the main processor and the supplemental processor when the IEEE 802.11 protocol is loaded onto the protocol processor. The operation procedure shown in Figure 7 is when the data packet matches with the terminal's address. When this occurs the supplemental processor assists the main processor by independently generating MAC response messages during the data reception procedure.

Figure 8 shows the paging procedure when the W-CDMA protocol is loaded onto the platform. The operation procedure shown in Figure 8 is when the destination of the paging message matches with the terminal. After periodically being activated by the power manager, the baseband modem wakes up the supplemental processor if the PICH bits indicate there will be a paging message on the PCH channel. Only the supplemental processor is activated while decoding a paging message. The main processor is not activated until the supplemental processor decodes the paging message completely.

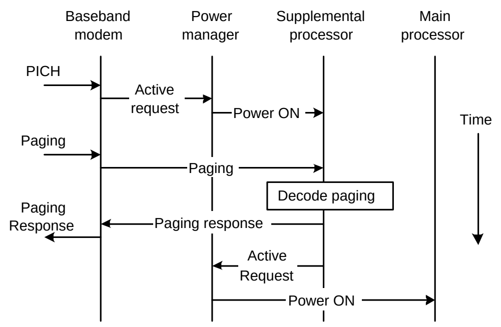

Figure 8: Paging message reception procedure while executing the W-CDMA protocol

5. EXPERIMENT AND ANALYSIS

5.1 Models for Experiment

5.1.1 Main Processor

As our main processor model we used the OPENRISC1200, a 32 bit open source RISC processor [3]. The features of OPENRISC are very similar to other commercial GPPs used in mobile devices except that it does not support power reduction techniques like DVS, DFS, and clock gating. However, this is not important because this model is only used for wake-up power measurement, where these techniques are not involved. We don't measure the dynamic power of the main processor, because it consumes far more power than the supplemental processor. The features of our HDL model for the main processor are as follows.

- 32 bit RISC processor
- Single issue 5-stage pipeline in-order machine
- Harvard architecture
- 32bit Hardware Multiplier
- 8K data cache with MMU
- 8K instruction cache with MMU

5.1.2 Supplemental Processor

Although we will show that a 32 bit processor is optimal for the supplemental processor in Section 5.2.2, we take as a base the architecture of a commercial 8 bit processor, Microchip's 16F84, in order to develop a power model of the supplemental processor [1]. It is used for a sensor network node that has similar requirements. We added a RNG, a hardware comparator, and a data memory interface for fast packet processing. Figure 9 shows the architecture of the supplemental processor model. The detailed configuration of the supplemental processor is as follows.

- 8 bit RISC architecture
- Single issue 2-stage pipeline in-order machine
- Harvard architecture
- 33 instructions
- 68 x 8 bit registers
- Hardware timer, interrupt handler, and clock gating
- Hardware random number generator
- Interface to data memory
- Hardware 48 bit comparator
- 4K x 14 bit program memory
- 4K x 8 bit dual-port state memory

5.1.3 Application Software

For this study we selected the power managment procedures for the IEEE 802.11 protocol, which is depicted in Figure 4, as the application model for our dynamic power measurements. This procedure was selected because it is currently one of the most complex and time constrained applications for packet reception.

The IEEE 802.11 application model is divided into two parts: an interrupt service routine and the main routine. In order to avoid an event loss, the interrupt service routine performs only a minimal number of operations such as hardware re-initialization. The main routine consists of an infinite loop. At the beginning of each iteration, the main routine checks the event flags. If there are no events, the main routine enters a clock-gated mode to save power by executing sleep instructions. If an interrupt is generated by the timer or the baseband modem, the main routine activates the clock signal and checks the event flags to identify the interrupt source. In the event of a packet reception, the main routine loads the packet header into the hardware comparator to analyze whether the packet matches with the terminal's address or the broadcast address. If the packet matches and is valid, the main routine performs the appropriate actions like generating a response message and informing the main processor of a packet reception.

5.2 Dynamic Power

Dynamic power is only consumed during instruction execution. To measure it we synthesized our hardware model using Synopsys' Design Compiler and extracted power results with PrimePower. Memory was created with Artisan's memory compiler. Figure 10 shows the experiment flow used for dynamic power measurement. We did not consider the power consumption due to interconnection wire because generally it is not dominating factor at 0.18u technology which we used for the experiment.

5.2.1 Power Consumption Profile of the Supplemental Processor

Figure 11 shows the power consumption profile of the supplemental processor when it executes the software model outlined at Section 5.1.3. We see that the blocks that consume the most powers in the supplemental processor are the timer and the program memory. The timer is power hungry because it has to continuously monitor when to wake-up. Reducing power consumption of the timer will have a noticeable impact on the supplemental processor's dynamic power. In addition, its program memory consumes a large fraction of power because it is relatively large compared to other blocks on the supplemental processor. Thus it is important to minimize the program size of the supplemental processor. This is help by the fact that the idle mode operations such as paging and location update are can be compactly coded, because they simply consist of header analysis of packets.

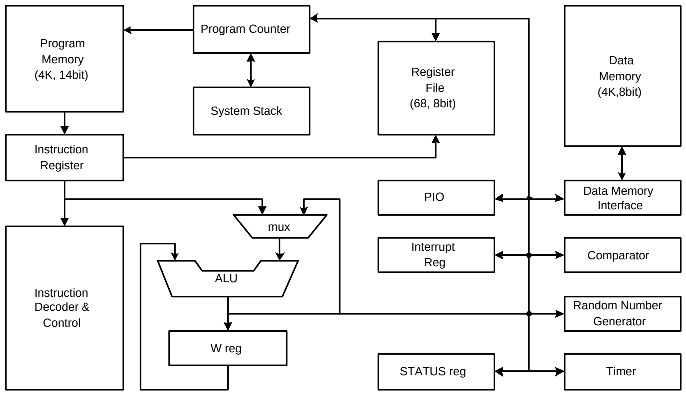

Figure 9: The architecture of the supplemental processor for the experiments

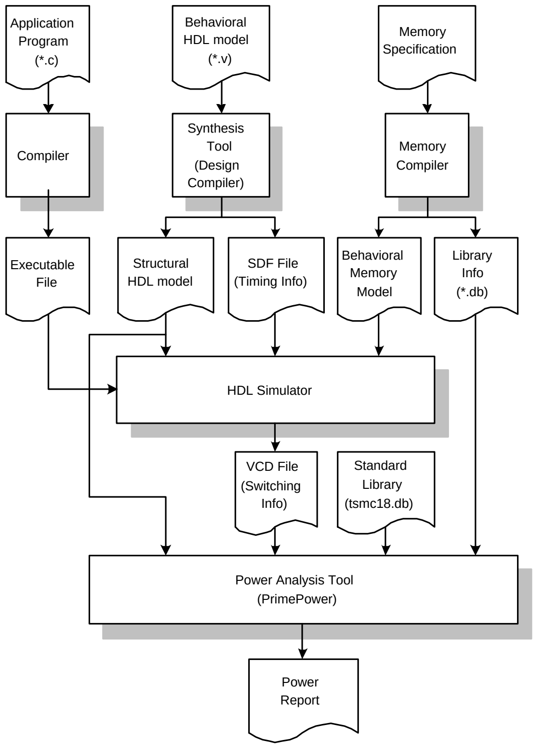
Application Program (*.c)
Behavioral HDL model (*.v)
Memory Specification
Compiler
Synthesis Tool (Design Compiler)
Memory Compiler
Executable File
Structural HDL model
SDF File (Timing Info)
Behavioral Memory Model
Library Info (*.db)
HDL Simulator
VCD File (Switching Info)
Standard Library (tsmc18.db)
Power Analysis Tool (PrimePower)
Power Report

Figure 10: Experiment flow to measure dynamic power

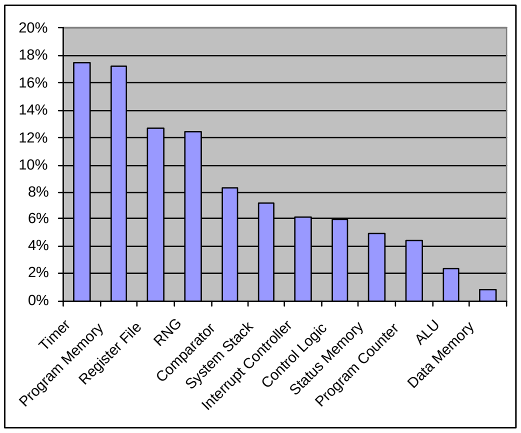

Figure 11: Power consumption profile of the supplemental processor

Minimum Operation Cycles	Minimum Frequency	Minimum Dynamic Power
360	136 Mhz	2.1 mW

Table 1: Minimum dynamic power of the supplemental processor

5.2.2 *Widening Datapath*

From the power consumption profile in Figure 11, we can predict the impact of widening data path from 8 to 32 bit. There are several inefficient points about the current hardware model for the supplemental processor that may be controlled by using 32 bits. The 16F84 chip, the baseline of the supplemental processor, was designed for a sensor network. An 8 bit data path, a large register file, and the absence of external memory are appropriate for this application domain. However, a wider datapath and a smaller register file are more appropriable for our use, because the MAC protocol requires frequent data memory access. Our proposed 16x32 bit register file is about the same size of the 68x8 bit register file normally in the 16F84 chip. This change has little effect on power consumption but a 32 bit ALU increases its power consumption from 2.5% to 10% of total. However, a 32 bit datapath allows to reduce the operating frequency almost to one quarter of the 8 bit version. The reduction in operating frequency causes a power reduction effect on all blocks of the supplemental processor.

5.2.3 *Minimum Dynamic Power of the Supplemental Processor*

The following equation shows the relation between operation frequency of the supplemental processor and its dy-

	Supplemental Processor	Main Processor
$A_{cell}(\mu m^2)$	4.17E4	2.2E5
$A_{inter}(\mu m^2)$	2E5	4.2E6
$E_{wake-up}(\mu J)$	0.11	6.04
$E_{dynamic}(\mu J)$	42.3	-
Ratio($E_{wake-up}/E_{dynamic}$)	0.26%	-

Table 2: Wake-up energy of the processors

namic power consumption.

$$P = \alpha f, \quad \text{where } \alpha = 1.6E\text{-}11 \tag{1}$$

In order to obtain the coefficient α in equation (1), we measured the dynamic power of the supplemental processor at 100 MHz.

To keep power low, it is important to find the minimum operating frequency that meets the processing time constraints. We executed the worst case software model outlined previously in Section 5.1.3 and found that we needed about 360 cycles to generate a MAC response. In addition there is a need to encrypt/decrypt the MAC packet. This is usually done with a separated hardware encryption/decryption engine. The implementation of an advanced encryption standard (AES) engine in [10] shows that ciphering will take about 320 cycles. If we assume that 5 usec are reserved for the physical layer, this leaves 5 usec for the MAC layer (recall, we require a response in 10 usec). Thus a total of 360+320 cycles must be completed in 5 usec. This yields a minimum operating frequency of (360+320)/5usec = 136 MHz. By substituting this result into Equation (1), we get about 2.1 mW as the minimum power consumption of the supplemental processor. These results are shown in Table 1.

5.3 Wakeup Power

The wake-up power is the power required for initial circuit power up. Because there is frequent power on/off operations during idle periods, we need to analyze the impact of the wake-up power to evaluate power efficiency. The wake-up energy is represented by the following equation where the C is the aggregated capacitance of whole circuit and the V is its operating voltage.

$$E_{wake-up} = CV^2 \tag{2}$$

If we assume a fixed operating voltage, the wake-up energy depends on the aggregated capacitance of a circuit. Thus the wake-up power analysis requires the derivation of the parasitic capacitance of a circuit. We assume the following to simplify the derivation of parasitic capacitance:

A1: The circuit is only built of two input NAND gates.

A2: The initial gate inputs are evenly distributed between 0 and 1.

A3: Half of the interconnection area is covered with metal wires.

A1 simplifies the derivation of an average parasitic capacitance per unit area, and means we can avoid the extraction of the parasitic capacitance of all library cells. A2 is required to determine the initial input value of two input NAND gates when we measure wake-up current. Finally, A3 allows us to estimate the effect of interconnection wires.

Based on A1, we extract the netlist and parasitic capacitance of a two input NAND gate from the TSMC18 standard library layout information. With the extracted information we perform Hspice simulations and derive the energy consumed for the power up of the NAND gate cell (E_{NAND2}). Next we calculate a gate count by the dividing of the total cell area (A_{cells}) obtained from the synthesis tool by the area of a two input NAND gate (A_{NAND2}). To analyze the impact of interconnection wires we use the parametric test results provided by MOSIS(C_{inter}) [2]. From the synthesis tool, we also extract the area of interconnect wires (A_{inter}). From A3 we can easily estimate the capacitance of interconnect wires and the energy consumed to charge up the interconnect wires. We combine these terms to calculate the energy for the system wakeup ($E_{wake-up}$) as follows:

$$E_{wake-up} = E_{NAND2}\frac{A_{cells}}{A_{NAND2}} + C_{inter}V^2 \tag{3}$$

Table 2 shows the wake-up energy of the main and supplemental processors and the parameter used in this calculation.

To compare dynamic energy ($E_{dynamic}$), we convert the dynamic power, shown in Table 1, into dynamic energy by multiplying by the wake-up period (20 ms) of the ATIM window. From this we see that the wake-up power has no significant impact on the consumption profile of the platform. The wake-up energy of the supplemental processor is about 0.26% of its dynamic energy. This implies that the power state of the supplemental processor can frequently be changed without increasing a significant power penalty. However the wake-up energy of the main processor is about 14% of the dynamic energy of the supplemental processor. Therefore the main processor should be activated only when necessary. Our separation into two asymmetric processors facilitates this.

5.4 Stand-by Time Analysis

Stand-by time is the time a terminal can operate without battery recharging. The stand-by time is determined by following terms: I_a, the current in the active period; T_a, the length of the active period; I_s, the current in the sleep period; T_s, the length of the sleep period; and $Q_{battery}$ the capacity of a battery. The term I_s consists of the leakage current of the platform, and I_a is the current flowing into circuits in the active period. Then the stand-by time of a terminal is represented by the following equation:

$$\text{Stand-by time} = Q_{battery}\frac{T_a + T_s}{I_aT_a + I_sT_s} \tag{4}$$

The term I_a is composed of several other terms: I_{syn}, the current of the frequency synthesizer; I_{trans}, the current of the receiver module of the transceiver; I_{base_RX}, the current of the receiver module of the baseband modem; and I_{proc}, the current of the supplemental processor:

$$I_a = I_{syn} + I_{trans\_RX} + I_{base\_RX} + I_{proc} \tag{5}$$

In our calculations we use the following parameters: For active and sleep durations, we use the common default values, T_a = 20 msec and T_s = 80 msec. The battery capacity is determined from the data sheet of a Panasonic battery, $Q_{battery}$ = 1035 mAh. The leakage current in the sleep period is assumed to be I_s = 0.1 mA. The parameters for

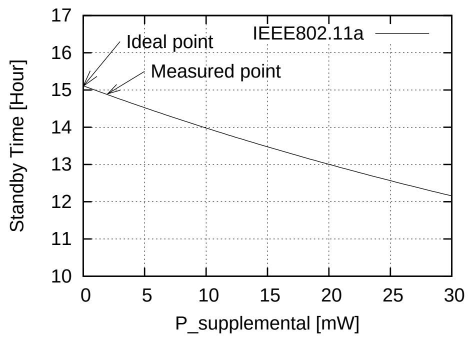

Figure 12: Stand-by time of the IEEE 802.11a terminal

the circuit power consumption in the active period are from Atheros Communication's 802.11a chip parameters, $I_{syn} = 72$ mA, $I_{trans\_RX} = 100$ mA and $I_{base\_RX} = 170$ mA [11].

Figure 12 shows the resulting relationship between the stand-by time of an IEEE802.11a terminal and the power consumption of the supplemental processor. The reason for choosing the IEEE 802.11a terminal for stand-by time analysis is that the power consumption of the supplemental processor is greatest for the IEEE 802.11 protocol.

5.5 Impact of Low Power Supplemental Processor

According to the dynamic power analysis result in Section 5.2.3, the average dynamic power of the supplemental processor is 2.1 mW, see Table 1. Looking this value up on x-axis of Figure 12 yields a stand-by time of about 14.9 hours for the IEEE 802.11a terminal - refer to the measured point of Figure 12. To compare with a power efficient ASIC based platform: assume an ideal case that consumes 0 watts for the MAC protocol processing. In this ideal case the stand-by time of the IEEE 802.11a terminal is about 15.1 hours - see the ideal point of Figure 12. The difference between the measured point and the ideal point is just 0.2 hours, because the power consumption of the supplemental processor is not a dominant source of power dissipation as compared to other blocks such as the baseband modem and transceiver. Thus the supplemental processor based on GPP is also a power efficient solution.

6. CONCLUSION

In this paper we proposed a flexible hardware platform for the upper layer protocols of an SDR terminal. Our analyses showed that a GPP, while flexible enough, is difficult to program given realtime constraints. As a solution we proposed an asymmetrical dual-processor platform comprised of two GPPs: a conventional main processor, and a simple low power supplemental processor. The supplemental processor provided the flexibility and realtime response capability with minimum power overhead. We validated the efficacy of the supplemental processor from a power and response time perspective using commercial CAD tools. The stand-by time analysis showed that ASICs are only marginally more power efficient. In summary the dual-processor platform meets the realtime deadlines of an SDR terminal at low power while maintaining programmability.

7. REFERENCES

[1] http://www.microchip.com.

[2] http://www.mosis.org.

[3] http://www.opencores.org.

[4] Jim Bohac. Flexible Chip Set Arms 802.11a/b/g WLANs. *Microwaves & RF*, May 2003.

[5] Hyok-Sung Choi and Hee-Chul Yun. Context Switching and IPC Performance Comparison between uClinux and Linux on the ARM9 based Processor. Technical report, Samsung Electronics, 2005.

[6] Lawrence T. Clark, Eric J. Hoffman, Jay Miller, Manish Biyani, Yuyun Liao, Stephen Strazdus, Michael Morrow, Kimberley E. Velarde, and Mark A. Yarch. An Embedded 32-b Microprocessor Core for Low-Power and High-Performance Applications. *IEEE Journal of Solid-State Circuits*, 36(11):1599–1608, November 2001.

[7] Virantha Ekanayake, Clinto Kelly IV, and Rajit Manohar. An Ultra Low-Power Processor for Sensor Networks. In *ASPLOS*, 2004.

[8] Toshio Fujisawa, Jun Hasegawa, Koji Tsuchie, Tatsuo Shiozawa, Tetsuya Fujita, Toshitada Saito, and Yasuo Unekawa. A Single-Chip 802.11a MAC/PHY with a 32-b RISC Processor. *IEEE Journal of Solid-State Circuits*, 38(11):2001–2009, November 2003.

[9] R. Kumar, Keith I. Farkas, Norman P. Jouppi, Parthasarathy Ranganathan, and Dean M. Tullsen. Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction. In *MICRO-36*, 2003.

[10] Henry Kuo, Ingrid Verbauwhede, and Patrick Schaumont. A 2.29 Gbits/sec, 56 mW Non-Pipelined Rijndael AES Encryption IC in a 1.8V, 0.18 um CMOS Technology. In *IEEE Custom Integrated Circuits Conference*, 2002.

[11] Teresa H. Meng, Bill McFarland, David Su, and John Thomson. Design and Implementation of an All-CMOS 802.11a Wireless LAN Chipset. *IEEE Communication Magazine*, 41(8):160–168, August 2003.

[12] Leyla Nazhandali, Bo Zhai, Javin Olson, Anna Reeves, Michael Minuth, Ryan Helfnd, Sanjay Pant, Todd Austin, and David Blaauw. Energy Optimization of Subthreshold-Voltage Sensor Network Procssors. In *ISCA*, 2005.

[13] Kevin J. Nowka, Gary D. Carpenter, Eric W. MacDonald, Hung C. Ngo, Bishop C. Brock, Koji I. Ishii, Tuyet Y. Nguyen, and Jeffrey L. Burns. A 32-bit PowerPC System-on-a-Chip With Support for Dynamic Voltage Scalaing and Dynamic Frequency Scaling.

[14] C. Michael Olsen and L. Alex Morrow. Multi-Processor Computer Systems Having Low Power Consumption. In *PACS*, 2003.

[15] Upkar Varshney and Radihika Jain. Issues in Emerging 4G Wireless Networks. *IEEE Computer*, June 2001.