

Reducing Idle Mode Power in Software Defined Radio Terminals

Hyunseok Lee, Trevor Mudge
Dept. of EECS
University of Michigan
{leehzz,tnm}@eecs.umich.edu

Chaitali Chakrabarti
Dept. of EE
Arizona State University
chaitali@asu.edu

ABSTRACT

In this paper, we propose a processor which is optimized for idle mode operation of a software defined radio (SDR) terminal. Since a SDR terminal spends most of its time in the idle mode, reducing the power consumption in this mode directly translates to longer terminal standby time. Workload analysis of idle mode operations of contemporary standards showed that these are dominated by FIR filtering, which can be easily parallelized. This analysis was used in the design of the idle mode processor. The key architectural components are an SIMD unit for the parallel computations that dominate the workload, a conventional scalar unit for the sequential computations, and a control unit which supports efficient data memory access and loop control. The idle mode processor was modeled with Verilog and synthesized using standard cells in 0.13 micron technology. It consumes about 9mW at 1.08V.

Categories and Subject Descriptors

C.14 [Processor Architecture]: Other Architecture Styles
– Mobile processors

General Terms

Design, Performance

Keywords

Software defined radio, SDR, Wireless terminal, Baseband processor, Idle mode, Low power, SIMD

1. INTRODUCTION

Software defined radio (SDR) is a wireless communication system whose function blocks are implemented by flexible software routines instead of fixed hardware, so that various wireless protocols can be easily supported on the same platform and future changes can be smoothly accommodated. In

this paper we consider the baseband processor, which is digital hardware that performs complex signal processing algorithms for communicating over unreliable wireless channels. The baseband processor deals with the most computationally demanding part of the wireless communication system and, as a result, is usually realized as an ASIC. Although the baseband processor can be used both at the basestation and the wireless terminal, we focus on the baseband processor in the wireless terminal. This makes power reduction in the baseband processor particularly important.

The operation of the wireless terminal can be classified into two modes: *active mode* when a wireless terminal actively transmits and receives user traffic, and *idle mode* when a wireless terminal waits passively to respond to communication requests from other terminals. Because the workload of the active mode is substantial, most previous research on SDR platforms has focused on the efficient support of the active mode operations. However, for the end user, one of the most desirable features of a wireless terminal is long battery life [1]. A significant portion of battery power is used in the idle mode. Although the energy dissipated by a single idle mode operation is very small, the aggregated effect is substantial, because a wireless terminal spends most of its time in the idle mode. In the case of W-CDMA terminal, the idle time can be as much as 99% [2][3].

The main topic of this paper is to minimize the idle mode power consumption of the baseband processor for SDR terminals without limiting programmability too much. We will show that this can be achieved, because the number of distinct signal processing kernels of the idle mode is limited.

The first step in our work is to identify the signal processing algorithms used in the idle mode and to characterize their computation patterns. We analyze the idle mode operation scenario of several contemporary standards, including GSM, GPRS, EDGE, IS-95, cdma2000, W-CDMA, IEEE 802.11/a/b/g, WiMax, and WiBro. In the workload analysis described in Section 2 we find that, although wireless terminals perform many operations in the idle mode, the majority falls into the class of finite impulse response (FIR) filtering which can be represented by $\sum_{i=0}^{L-1} c_i \cdot x[i+n]$. The absence of data dependency between the terms means that the multiplications can easily be parallelized. There still exist some sequential computations in the idle mode operation; however, their impact is not as significant.

Based on these observations, we propose an idle mode processor which consists of three sub-units: a single instruction multiple data (SIMD) unit, a scalar unit, and a control unit. The SIMD unit handles the parallel workload, namely

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISLPED'06, October 4–6, 2006, Tegernsee, Germany.
Copyright 2006 ACM 1-59593-462-6/06/0010 ...\$5.00.

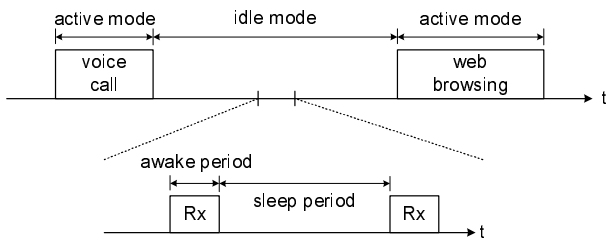


Figure 1: The active and idle operation modes of wireless terminal, and the awake and sleep periods in the idle mode

the FIR filter; the scalar unit handles the sequential workload; and the control unit assists the SIMD and scalar units with automatic data memory address generation and loop control. The operations in the FIR filter can be efficiently mapped onto the SIMD unit allowing for significant power savings – parallelization makes possible lower operation frequency and slower but more power efficient circuits. The proposed idle mode processor can also be used in the active mode, because the key idle mode computation kernels are also among the computationally-intensive kernels used in the active mode.

To validate our idea, we implement a hardware model of the proposed architecture using Verilog and synthesize it using 0.13 micron technology. This model is used as an input to Synopsys’ PrimePower, a commercial power evaluation tool. The experiments show that the proposed idle mode processor consumes about 9mW at 1.08V when performing the idle mode operations. This is more than a 80% dynamic power reduction compared to state-of-the-art SDR processors optimized for the active mode [4]. Such significant power reduction is due to the lower operation frequency made possible by the simplified SIMD architecture. Further power reduction in low activity periods can be achieved by using more advanced silicon process technology and techniques such as clock gating.

This paper is not the first attempt to optimize the power consumption of hand held devices when they are in a low activity mode. There are several low power general purpose processors targeted for hand held devices. Examples are personal data assistants based on Intel’s Xscale [5] and IBM’s PowerPC 405 [6]. However, the application domain of these processors is quite different from that of the baseband processor. While there have been a number of projects that explore SDR platforms [7][8][9][10][11], these efforts are primarily focussed on the active mode, and ignore the idle mode. The contribution of this paper is to characterize the operation of a SDR terminal in the idle mode, and to propose an architecture that is optimized for idle mode operation and can be used as part of the baseband processor of an SDR terminal.

2. WORKLOAD ANALYSIS

2.1 Operation Modes of Wireless Terminal

Wireless terminals change their operation state according to use activity as depicted in Figure 1. When there are active user applications, wireless terminals employ all of their functionality to support high data rate communications (*active mode*). However, even when there exist no

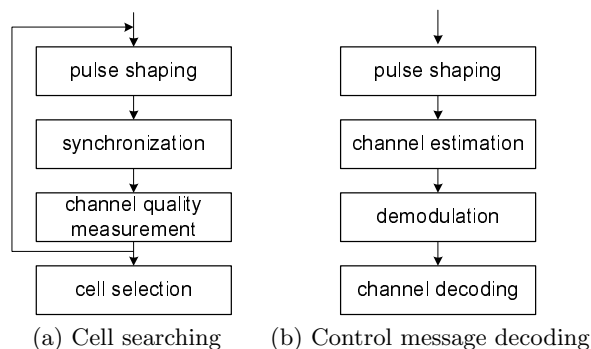


Figure 2: Generalized idle mode operation flow of wireless terminals in basestation based networks

active user applications, wireless terminals can not turn off completely, because requests from other terminals may be received at unpredictable times. So, wireless terminals activate a subset of their communication components to receive the requests (*idle mode*). In the idle mode, the operation is intermittent. Instead of continuously monitoring for signals, terminals completely power off their receiver (*sleep period*) and activate it only for short pre-scheduled periods (*awake period*).

2.2 Operations in Idle Mode

In this section, we discuss in detail the idle mode operation of wireless terminals. We consider idle mode operation for two cases: a wireless network with basestations and a wireless network without basestations (ad hoc network).

2.2.1 Operation of Network with Basestation

Most wireless networks such as GSM, GPRS, EDGE, IS-95, cdma2000, and W-CDMA use basestation. In such networks, idle mode operations consist of two steps, cell searching and control message decoding, as shown in Figure 2. In the cell searching step, the terminal selects basestations that provide best quality channel among candidates. In the control message decoding step, the terminals receive and decode control messages from selected basestations.

Cell searching: As shown in Figure 2(a), the cell searching step consists of four procedures: pulse shape filtering, synchronization, channel quality measurement, and cell selection. Pulse shape filtering is a common low pass filtering operation used in all wireless terminals. It is typically implemented with an FIR filter. The synchronization procedures of most wireless networks have considerable similarity: a basestation broadcasts a reference signal containing a synchronization code, the terminals extract timing and frequency information from this signal by matched filtering of received signal with the synchronization code. After the synchronization, terminals measure the quality of channel provided by the synchronized basestation. Channel quality is measured by the fidelity of the decoded reference signal. As shown in Figure 2(a), the terminals iteratively perform the pulse shaping, synchronization and channel quality measurement procedures against all candidate basestations. Cell selection is to select a set of basestations that provide best quality channel.

Control message decoding: After the cell searching, the pulse shaped signal is passed to the channel estimator which

estimates the wireless channel characteristic. The estimated characteristics are used to demodulate the received signal. Demodulation is to reconstruct the most probable information sequence from the received signal. Terminals of TDMA networks typically use the FIR filter and a Viterbi decoder for the channel estimation and demodulation. Terminals of CDMA networks perform matched filtering for the channel estimation. The demodulated signal is processed by the channel decoder which performs forward error correction. Turbo decoders and Viterbi's decoders are typical examples of channel decoder. The decoded result is forwarded to upper layers. Although the channel estimation and channel decoding procedures are the ones with heaviest workloads in the active mode, their workload is not substantial in the idle mode because the control message decoding is performed only once while the synchronization and channel quality measurement procedures are performed multiple times against all candidate basestations. Thus, the idle mode workload of basestation based networks is dominated by the synchronization and channel quality measurement procedures.

2.2.2 Operation of Ad Hoc Network

The operation flow shown in Figure 2 is not valid for ad hoc networks. To avoid severe power penalty, a wireless terminal in an ad hoc network does not continuously broadcast the reference signal to synchronize all terminals within a network. Instead, terminals attach a preamble sequence at the header of all transmitted frames. In order to detect the transmission of a new frame, terminals compute the correlation between the received signal and its delayed signal. This type of computation is called a 'sliding window' and its detailed characteristics will be discussed in the followed subsection. Wireless local area networks (WLAN) such as IEEE 802.11/a/b/g sometimes operate in the ad hoc mode.

2.3 Key Computation Patterns in Idle Mode

There are two key kernels that dominate the idle mode operations: FIR filtering and sliding window. For instance, the FIR filter operation occupies about 80% of the idle mode workload of W-CDMA terminal [12]. It follows that the power efficiency of the idle mode processor is dominated by the power efficiency of these kernel computations.

2.3.1 FIR Filter

The computation pattern of the FIR filter which is used for the pulse shaping and the synchronization can be represented by following equation:

$$y[n] = \sum_{i=0}^{L-1} c_i \cdot x[i+n] \quad (1)$$

where $x[n]$ is the input signal, c_i is the filter coefficient, and L is the filter length. The coefficient of the FIR filter is typically an arbitrary number. However, in matched filtering with the synchronization code sequence which consists of 1 or -1, the multiplications shown in Equation (1) can be simplified into conditional complements as follows:

$$c_i \cdot x[i+n] = \begin{cases} -x[i+n], & \text{if } c_i = -1 \\ x[i+n], & \text{otherwise} \end{cases} \quad (2)$$

As shown in Figure 3, there exist two methods to implement the computation pattern shown in Equation (1):

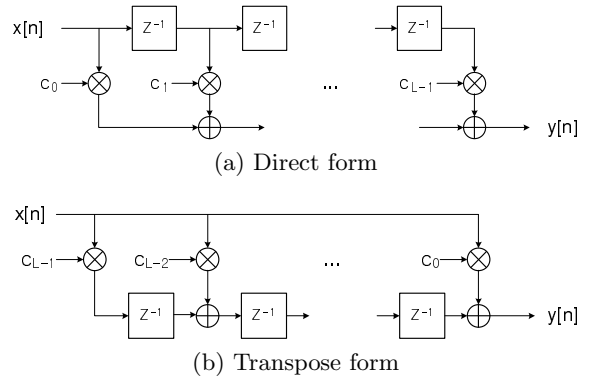


Figure 3: Two implementation methods for FIR filter

direct form and transpose form. Although they are functionally identical, these two implementation methods have different power costs.

There exist a lot of parallelism in the FIR filter. In Equation (1), the L multiplications for one output $y[n]$ can be performed in parallel. Furthermore, the multiplications for other outputs $y[n+1], \dots, y[n+k]$ also can be done in parallel if resources are available. Despite such high parallelism, the FIR filter requires only one new input data, $x[n]$, to produce a new output, $y[n]$. The characteristics of the FIR filter vary according to the type of wireless network. The range of the filter order L is 30~300 taps. These can be broken into 32-wide parallel chunks for SIMD processing. The input data, $x[n]$, and filter coefficient, c_i , are represented by 1~16 bits.

2.3.2 Sliding Window

The computation pattern of the sliding window operation which is used for the frame detection can be represented as follows:

$$y[n] = \sum_{i=0}^{L-1} P_{i,n} \quad (3)$$

where $P_{i,n} = x[i+n] \cdot x[i+n-D]$, $x[n]$ is the input signal at time n , L is the correlation length, and D is the delay between input signals. It is equivalent to auto-correlation of $x[n]$ with delay D and can be implemented with L multiplications and $L-1$ additions per output. However, the computation shown in Equation (3) can be further simplified due to the following relation between outputs:

$$y[n] = y[n-1] - P_{0,n-1} + P_{L-1,n} \quad (4)$$

Because we can reuse the previously computed $P_{0,n-1}$, the operation shown in Equation (4) additionally requires only one multiplication, addition, and subtraction for the generation of the next output. Therefore, the sliding window operation can be classified as a scalar workload and implemented by the scalar unit.

2.4 Processing Time of Idle Mode

In addition to the limited number of computation patterns that have to be supported, the idle mode operation also has more relaxed processing time requirements compared to that of the active mode. To avoid buffer overflow, a baseband processor must finish the operations on the current frame

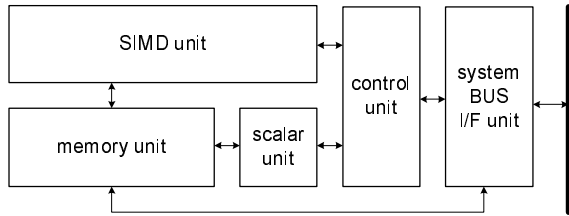


Figure 4: Architecture of the idle mode processor

```
for(i=0; i<Max.Sample; i++) {
  for(j=0; j<Filter.Len; j++) y[i] += x[j+i]*c[j];
}
```

(a) C routine for the FIR filter with direct form

```
Loop: shift.v   VR1 ← VR1, (AR1)
      mul.v    VACC ← VR2, VR1
      reduction.v (AR2) ← VACC
      inc     AR1, #1
      inc     AR2, #1
      dec     SR1, #1
      jnz    SR1, Loop
```

(b) Pseudo assembly routine for SIMD implementation

```
Loop: shift.v   VR1 ← VR1, (AR1++)
      mul.v    VACC ← VR2, VR1
      reduction.v (AR2++) ← VACC | djnz SR1, Loop
```

(c) Pseudo assembly routine after applying automatic address generation and loop control

```
Loop: reduction.v (AR2++)
      ←{mul.v VR2, {shift.v VR1, (AR1++)}} | djnz SR1, Loop
```

(d) Pseudo assembly routine after cascading arithmetic units

Figure 5: FIR filter routine optimized for the SIMD implementation, automatic address generation and loop control, and cascading arithmetic units

before the next frame arrives. In the idle mode, the inter-frame arrival time is longer than that of the active mode, by at least an order of magnitude. Thus, it is important for the idle mode processor to exploit the longer processing time for power reduction.

3. PROCESSOR DESIGN

3.1 Overall Processor Architecture

At the top level, the idle mode processor consists of five units as shown in Figure 4: SIMD unit, scalar unit, control unit, memory unit, and system BUS interface unit. The SIMD unit implements the FIR filter. The scalar unit implements unparallelizable workloads, including the sliding window. The memory unit stores operation data and program. The system BUS interface unit is used to communicate with other entities in the system, such as the analog front-end. The control unit manages all the units; it includes hardware address generators and hardware loop counter in addition to conventional program counter, instruction decoder, and control logic.

In order to explain the following high level design decisions, we represent the FIR filter operation in ‘C’ and pseudo assembly routines. As shown in Figure 5(a), the operation

of FIR filter consists of two loops: the inner loop consists of operations for one output and the outer loop represents the multiple iterations of the filter operation. Through the use of the proposed hardware features, the ‘C’ routine shown in Figure 5(a) can be simplified into a single line of assembly routine as shown in Figure 5(d).

SIMD unit: The SIMD architecture is appropriate for the inner loop operation shown in Figure 5(a). The inner loop operation consists of identical computations with regular operand access patterns. The SIMD architecture is power efficient; the parallel execution of the computations of the inner loop helps reduce operations frequency and this directly translates to dynamic power reduction. The hardware complexity needed to provide operands to the SIMD datapath is low. This is because the inner loop operation is a single input and single output system, which can be implemented by a low power single port memory. Figure 5(b) describes the corresponding pseudo assembly routine that can be mapped to the SIMD architecture.

Control unit: The hardware address generators and loop controller in the control unit maximize the utilization of SIMD unit. As shown in Figure 5(b), only three out of the seven instructions are SIMD instructions. The other scalar instructions are related to address generation and loop control. Fortunately, the FIR filter operation has regular data memory access pattern. So, it is possible to automate the address generation without significant overhead. Because the FIR filter is single input – single output systems, two address generation units are sufficient, one for input data and the other for output data. The two loop control related instructions can also be automated easily by using decrementing counter and zero detection logic. The result of deploying automatic address generation and loop control is shown in Figure 5(c). In this routine, ‘djnz’ stands for decrease and then jump if the decrement result is not zero.

Scalar unit: To support the sequential workload, the idle mode processor consists of a scalar unit as well. Because the workload assigned to the scalar unit is control intensive and also includes computations such as multiplication, division, and logical operations, a conventional low power general purpose processor is a suitable choice. A multiplication and accumulation (MAC) unit is useful for minimizing the operation cycle of the scalar unit.

Memory unit: The memory unit consists of data and instruction memories. We estimated that 100 Kbytes data memory and 100 Kbytes instruction memory are sufficient for idle mode operation. To reduce power consumption, memories are divided into smaller sub-banks. Such a structure significantly reduces the dynamic power, because only one sub-bank is activated during read or write. Although sub-banking demands more silicon area, we use this scheme because power reduction is crucial in the idle mode processor. In addition, for further power reduction, each sub-bank is implemented with a single port memory. Data is distributed among the sub-banks in a way such that data read and write can be done in parallel.

System BUS interface unit: Because the idle mode processor is part of the wireless terminal, an interface with the system BUS is essential. According to our previous work [12], a low speed BUS is enough for the baseband processor, because communication rate between kernel algorithms is quite low. A client logic of advanced micropro-

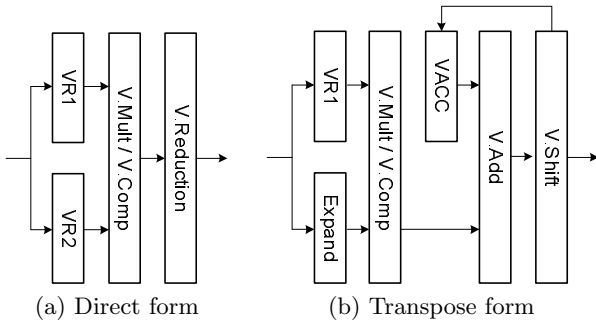


Figure 6: Two implementation methods of SIMD datapath

cessor bus architecture (AMBA) bus is a typical example of a system BUS interface unit.

3.2 Detailed Design of SIMD Datapath

Direct form SIMD datapath: As discussed before, the FIR filter can be implemented in two ways. Figure 6 depicts the detailed structures of the SIMD datapath based on both the direct form and the transpose form. According to the dynamic power analysis in Section 4, the direct form SIMD datapath has about 20% better power performance. Thus, the SIMD datapath is implemented with the direct form.

Cascading arithmetic units: As shown in Figure 6(a), the second design decision on the SIMD datapath is to cascade arithmetic units such as the adder and multiplier so that the temporary results do not have to be stored in a register file. This is motivated by the fact that accessing the register file is power expensive. Generally cascading of the arithmetic units limits system flexibility, a feature which is important for the SDR system. However, the limited flexibility is not a problem in the idle mode processor, because the SIMD datapath with cascaded function units is flexible enough to cope with many varieties of FIR filter operations, which can appear at the idle mode operation of wireless protocols.

Pipelining: The SIMD datapath has three pipeline stages: read, execution-1, and execution-2. The idle mode workload is suitable for pipelining, because it can fully utilize the pipeline by repeating identical computations many times. For instance, in Figure 5(a), the inner loop operation is repeated for each input sample (`Max_Sample` times). Due to the absence of data dependencies, no data forwarding path is required. So, the SIMD datapath can be pipelined by simply inserting flip flops between pipeline stages. Although these flip flops contribute additional power, this can not be more than the power reduction achieved from operating at a lower frequency and by the preventing of glitch propagation.

4. EXPERIMENT AND ANALYSIS

Methodology: We validated the functional correctness of the proposed idle mode processor architecture with a Verilog model. For power evaluation, we synthesized it using the TSMC13 library based on 0.13 micron technology. We estimated the dynamic power of the proposed processor at the gate level with the help of the commercial power evalu-

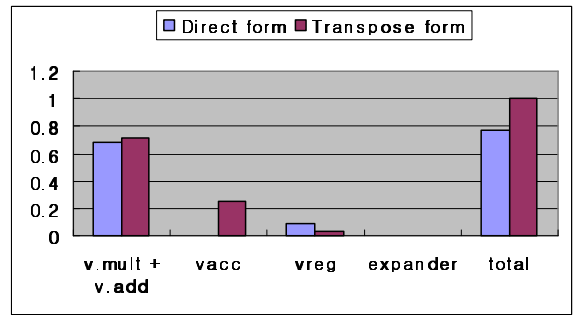


Figure 7: Normalized average dynamic power of SIMD datapaths: direct form vs. transpose form

ation tool, Synopsys' PrimePower. For memory generation, we used Artisan's memory compiler.

System configuration for experiment: We assumed that the SIMD width is 32. We used this number for compatibility with our SDR processor for active mode [4]. In general, the optimal SIMD width depends on design constraints such as silicon area and the type of silicon processing technology. Also, the W-CDMA was chosen as the application for the experiment since it has the most complex idle mode operation scenario. The implemented idle mode operation of the W-CDMA terminal was based on [13].

Minimum operation frequency estimation: In the measurement of dynamic power, the allowed minimum operation frequency is an important parameter. Although the length of the awake period is one of the design parameters, we assume it to be 30 msec: 27 msec for the cell searching and 3 msec for the control message decoding. From the behavioral model, we extracted the cycle count for idle mode operation. We used the allowed operation time and the cycle count to calculate the minimal operation frequency of the idle mode processor. For our design, the minimum frequency was about 36Mhz but we assumed it to be 50Mhz (adding about a 30% margin) for the dynamic power calculations.

Direct form vs. transpose form: Figure 7 shows the normalized average dynamic power of the SIMD datapaths based on the direct form and transpose form. It shows that the direct form implementation is about 20% more power efficient. This power efficiency comes from the absence of a vector accumulator (VACC) which stores the partial operation results in the direct form implementation. As show in Figure 3, the direct form stores input data having lower data precision whereas the transpose form stores partial computation results with higher data precision. Because a storage element requires more power compared to other logic elements, the VACC results in the transpose form SIMD datapath consuming higher power.

System dynamic power: Table 1 shows the dynamic power consumption of the idle mode processor. The most power consumer is the data memory. Although we applied memory sub-banking to reduce power, the data memory still dissipates about 43% of the total power. This is because the data memory is heavily accessed in most operations of the idle mode processor. The power consumption of the instruction memory is about one fifth the amount of the data memory, because it is only accessed for instruction reads by the control unit. The SIMD unit still consumes significant

Units	Subblocks	Peak Power (mW)	Average Power (mW)	Area (%)
SIMD	V.Mult	13.68	1.80	10.49
	V.Comp	2.24	0.24	1.96
	V.Reduction	3.80	1.21	2.31
	V.Reggs	0.90	0.38	1.96
Scalar	ALU	1.21	0.70	0.83
	Regs	0.53	0.31	0.07
Control	Address gen.	0.16	0.13	0.24
	Loop control	0.12	0.12	0.09
	Ctrl. logic	0.22	0.13	0.52
Memory	I-mem	0.85	0.49	37.88
	D-mem	3.93	3.32	43.29
BUS I/F	BUS I/F	0.16	0.04	0.02
Total	(rounded)	—	9mW	22mm ²

Table 1: Idle mode processor: Dynamic power @ (50Mhz, 1.08V) of each block during idle mode operation for the W-CDMA terminal, and area of each block

power (41%) even though we designed it with several low power techniques which are optimal for the idle mode operations. Thus, if we made the SIMD unit more programmable to support the algorithms in the active mode, the energy efficiency of the idle mode processor will be degraded substantially. Although the peak power of the vector multiplier is very high, its average is only 20% of total power. This is because the activation time of the vector multiplier is 10% of total operation time. Note that the vector multiplier only participates in the pulse shaping operation. The power overhead of the vector reduction logic is almost equivalent to that of the vector multiplier in average, because it is active throughout entire vector operation including the pulse shaping and the synchronization. The dynamic power of the scalar, control, and system BUS interface units is not substantial, approximately 16% of total power.

Table 1 also describes the area of each component. The scalar unit, the control unit, and the system BUS interface unit occupy negligible area (2.2%) compared to the SIMD unit (16.7%). However, the majority of the area (81%) is occupied by memories. Note that their sizes are likely to be determined by the overall SDR operation and not just the idle mode operation.

Further power reduction: The dynamic control of operand precision is an effective way for reducing power. This is because the operand precision varies across different wireless protocols even for the same operation. The dynamic control of operand precision directly affects the power consumption of dominating blocks. Second, use of low power implementation would result in additional power reduction. The current implementation only relies on a synthesis tool and memory compiler. Finally, synthesis of the processor in 90nm technology will help in achieving further power reduction.

5. CONCLUSION

In this paper we proposed a hardware platform for reducing the power consumption during idle mode operation of a SDR terminal. Our workload analysis showed that the idle

mode operation consists of a substantial amount of computations with simple SIMD parallelism, and a much smaller amount of control intensive sequential computation. Furthermore, although a wide variety of operations are performed in the idle mode, two computation kernels, namely, FIR filtering and sliding window, account for 80% of the computations in the idle mode. Based on this analysis, we proposed an architecture for the baseband processor that consists of three sub-units: 1) an SIMD unit for the parallel workload; 2) a scalar unit for the sequential workload; and 3) a control unit having hardware address generation and loop control. The use of a specialized SIMD unit for the dominant computations, and a fully programmable scalar unit for control intensive sequential operations resulted in a power efficient architecture for the idle mode processor. We estimated that the architecture, when synthesized in 0.13 micron technology, could consume about 9mW at 1.08V.

6. ACKNOWLEDGEMENT

This work was supported by Samsung Electronics and NSF-ITR CCR-0325898 and CCR-0325761.

7. REFERENCES

- [1] John Walko. Mobile phone users demand decent batteries. *EETimes*, Sep. 2005.
- [2] UMTS Forum. UMTS/IMT-2000 spectrum. *Technical Report* 6, 1998.
- [3] UMTS Forum. 3G offered traffic characteristics. *Technical Report* 33, 2003.
- [4] Y. Lin et al. SODA: A low-power architecture for software radio. In *International Symposium on Computer Architecture*, pages 89–101, June 2006.
- [5] L. Clark et al. An embedded 32-b microprocessor core for low-power and high-performance applications. *IEEE Journal of Solid-State Circuits*, 36(11):1599–1608, Nov. 2001.
- [6] K. Nowka et al. A 32-bit PowerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling. *IEEE Journal of Solid-State Circuits*, 37(11):1441–1447, Nov. 2002.
- [7] J. Glossner et al. A software defined communications baseband design. *IEEE Communication Magazine*, 41(1):120–128, Jan. 2003.
- [8] Simon Knowles. The SoC future is soft. In *IEE Cambridge Branch Seminar*, Dec. 2005.
- [9] A. Duller et al. Parallel processing — the picoChip way! In *Communicating Processing Architectures*, pages 125–138, Sep. 2003.
- [10] B. Mei et al. Architecture exploration for a reconfigurable architecture template. *IEEE Design and Test of Computers*, 22(2):90–101, March/April 2005.
- [11] C. Berkel et al. Vector processing as an enabler for software-defined radio in handsets from 3G+WLAN onwards. In *Software Defined Radio Technical Conference*, Nov. 2004.
- [12] H. Lee et al. Software defined radio — A high performance embedded challenge. In *High Performance Embedded Architectures and Compilers*, pages 6–26, Nov. 2005.
- [13] Y. Wang et al. Cell search in W-CDMA. *IEEE Journal on Selected Areas in Communications*, 18(8):1470–1482, Aug. 2000.