H. Lee, Y. Lin, Y. Harel, M. Woh, S. Mahlke, T. Mudge, K. Flautner.
1st International Conference on High Performance Embedded Architectures
and Compilers. November, 2005. Barcelona, Spain.

Software Defined Radio – A High Performance Embedded Challenge

Hyunseok Lee, Yuan Lin, Yoav Harel, Mark Woh, Scott Mahlke,
Trevor Mudge[1], and Krisztian Flautner[2]

[1] Advanced Computer Architecture Laboratory,
Electrical Engineering and Computer Science Department,
University of Michigan, 1301 Beal Ave. Ann Arbor, MI 48105-2122
{leehzz, linyz, yoavh, mwoh, mahlke, tnm}@eecs.umich.edu
[2] ARM Ltd. 110 Fullbourn Road, Cambridge, UK CB1 9NJ
Krisztian.flautner@arm.com

Abstract. Wireless communication is one of the most computationally demanding workloads. It is performed by mobile terminals ("cell phones") and must be accomplished by a small battery powered system. An important goal of the wireless industry is to develop hardware platforms that can support multiple protocols implemented in software (software defined radio) to support seamless end-user service over a variety of wireless networks. An equally important goal is to provide higher and higher data rates. This paper focuses on a study of the wideband code division multiple access protocol, which is one of the dominant third generation wireless standards. We have chosen it as a representative protocol. We provide a detailed analysis of computation and processing requirements of the core algorithms along with the interactions between the components. The goal of this paper is to describe the computational characteristics of this protocol to the computer architecture community, and to provide a high-level analysis of the architectural implications to illustrate one of the protocols that would need to be accommodated in a programmable platform for software defined radio. The computation demands and power limitations of approximately 60 Gops and 100∼300 mW, place extremely challenging goals on such a system. Several of the key features of wideband code division multiple access protocol that can be exploited in the architecture include high degrees of vector and task parallelism, small memory footprints for both data and instructions, limited need for complex arithmetic functions such as multiplication, and a highly variable processing load that provides the opportunity to dynamically scale voltage and frequency.

1 Introduction

Hand held wireless devices are becoming pervasive. These devices represent a convergence of many disparate features, including wireless communication, real-time multimedia, and interactive applications, into a single platform. One of the most difficult challenges is to create the embedded computing systems for these

devices that can sustain the needed performance levels, while at the same time operate within a highly constrained power budget to achieve satisfactory battery lifetimes. These computing systems need to be capable of supercomputer level performance levels with estimated performance levels of more than 60 Gops, while having a total power budget of about 100∼300 mW. The current generation of microprocessors and DSPs are not capable of meeting these performance and power requirements. The term "mobile supercomputer" has been used to describe such platforms [1].

In this work, we focus on the wireless communication aspect of hand held devices. Wireless communication is one of the most computationally intense workloads that is driven by the demand for higher and higher data rates. To support seamless service between various wireless networks, there is a high demand for a common hardware platform that can support multiple protocols implemented in software, generally referred to as software defined radio (SDR) [2]. A fundamental conflict exists when defining a computing platform for SDR, because performance, power, and flexibility are conflicting goals. At one extreme, which maximizes flexibility, are general purpose processors, where algorithms can be defined in high-level languages. At the other extreme, which maximizes performance and minimizes power, are application specific integrated circuits (ASIC). ASICs are hardwired solutions that offer almost no flexibility, but are the standard for current platforms.

Our goal, shared by others in the field, is to design and develop a programmable "mobile supercomputer" for SDR. It is first necessary to develop an understanding of the underlying requirements and computation characteristics of wireless protocols. The majority of the computation occurs at the physical layer of protocols, where the focus is signal processing. Traditionally, kernels corresponding to the major components, such as filters and decoders, are identified. Design alternatives for these are then evaluated on workloads targeted to their specific function. This approach has the advantage of dealing with a small amount of code. However, we have found that the interaction between tasks in SDR has a significant impact on the hardware architecture. This occurs because the physical layer is a combination of algorithms with different complexities and processing time requirements. For example, high computation tasks that run for a long period of time can often be disturbed by small tasks. Further, these small tasks have hard real-time deadlines, thus they must be given high priority. As a result, we believe it is necessary to explore the whole physical layer operation with a complete model.

From the many wireless protocols, we have selected the wideband code division multiple access (W-CDMA) protocol as a representative wireless workload to illustrate the operation of wireless communication systems. W-CDMA system is one of leading third generation wireless communication networks where the goal is multimedia service including video telephony on a wireless link [3]. W-CDMA improves over earlier cellular networks by increasing the data rate from 64 Kbps to 2 Mbps. Additionally, W-CDMA unifies a single service link for both voice and packet data, in contrast to previous generations which support only

one service. In order to study the requirements in more detail, we have developed a full C implementation of the W-CDMA physical layer to serve as the basis for our study. The implementation can be executed on a Linux workstation and thus studied with conventional architectural tools.

In this paper, we provide a detailed description and analysis of the W-CDMA physical layer. The fundamental computation patterns and processing time requirements of core algorithms are analyzed, along with the interactions between them. We also study the implications of the processing requirements on potential architectural decisions. One of the major keys to achieving the challenging power and performance goals is exploiting parallelism present in the computation, especially vector and task-level parallelism. This is balanced by a number of smaller real-time tasks that are more sequential in nature. Thus, it is important to consider both extremes and define an architecture capable of handling diverse types of processing.

The design of fully programmable architectures for W-CDMA is a difficult challenge faced by the industry. To date, no such design exists and thus serves as motivation for our analysis. Current DSP solutions, such as the TI TMS320C5XXX, have included specialized instructions, such as a compare-select instruction, designed specifically for wireless protocols. Further, there are many announced multiprocessor DSP systems that are designed specially for SDR. Some examples include the Sandblaster processor [4], the MorphoSys processor [5], and the 3plus1 processor [6]. However, none has yet to provide a fully programmable solution that could be programmed for other protocols and that satisfies both real-time W-CDMA performance requirement as well as being competitive with ASICs in power consumption. Some of these solutions, like MorphoSys processors, are aimed at basestations, where the power requirements are less stringent. To meet the W-CDMA processing requirements, many of these programmable processors also require ASIC accelerators for the most computationally demanding portions of the protocol.

2 W-CDMA protocol

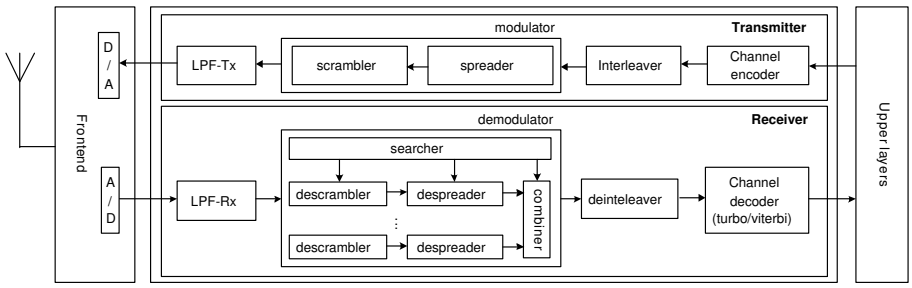

Fig. 1. High level block diagram of the physical layer operation of a W-CDMA terminal

The protocol stack of the W-CDMA system consists of several layers and each layer provides an unique function in the system. According to the computation characteristics, we can group the protocol layers into two parts: the physical layer and upper layers. The physical layer, which is placed at the bottom of the protocol stack, is responsible for signal transmission over an unreliable wireless link. To overcome noise that the environment introduces on the wireless link, the physical layer performs many computation intensive signal processing algorithms such as matched filtering and forward error correction. Meanwhile, upper layer protocols cover the control signaling required to operate within a wireless network. For example, the medium access control (MAC) layer provides a scheme for resolving resource contention between multiple terminals.

Although SDR encompasses all protocols layers, we only focus on the physical layer due to its computation and power importance compared to other layers. The operation of the physical layer is realized by both digital and analog circuits. Because the operation frequency of analog frontend circuits reaches GHz level, it is infeasible to replace the analog frontend circuits with programmable digital logic with current circuit technology. Thus, we further narrow down our focus to the physical layer that is realized with digital circuits. Figure 1 shows a high level block diagram of the physical layer operation of the W-CDMA terminal.

To aid in explanation, we define the following sets: a set of binary numbers with dual polarities, $\mathbf{B} = \{-1, 1\}$ [3]; a set of complex binary numbers, $\mathbf{B^C} = \{a + jb \mid a, b \in \mathbf{B}\}$; a set of m bit fixed point numbers, $\mathbf{I} = \{a \mid a, m \text{ are integers and, } -2^{m-1} < a \leq 2^{m-1}\}$; and a set of complex numbers represented by two m bit fixed point numbers, $\mathbf{I^C} = \{a + jb \mid a, b \in \mathbf{I}\}$.

Channel Encoder and Decoder The role of a channel encoder and decoder is for error correction. The channel encoder in a transmitter adds systematic redundancy into the source information, and the channel decoder in a receiver corrects errors within the received information by exploiting the systematic regularity of the redundant information. The W-CDMA physical layer uses two kinds of channel coding schemes: **convolutional codes** [7] and **turbo codes** [8]. The detailed description of the channel codes for the W-CDMA physical layer is in [9]. The encoders for both codes are simple enough to be implemented with several flip-flops and exclusive OR gates. However, the decoders for these codes are highly complex because their operation is to find a maximum likely code sequence from the received noisy signal.

Among many possible methods, our implementations for the channel decoders are based on a soft output Viterbi algorithm (SOVA), because of its lower computational complexity and the fact that it only shows a slight performance degradation compared to other methods. The difference between a conventional Viterbi algorithm and the SOVA is the use of "soft" numbers in the SOVA. If a soft number is used, each bit plus noise in the received sequence is quantized as a fixed point number with higher precision (e.g. 4 bits). Although it requires more

[3] In conventional binary notation $-1 \equiv 1$ and $\mathbf{1} \equiv 0$

computation power and memory, the use of soft number is necessary in most practical W-CDMA receivers to provide high fidelity signal processing gain.

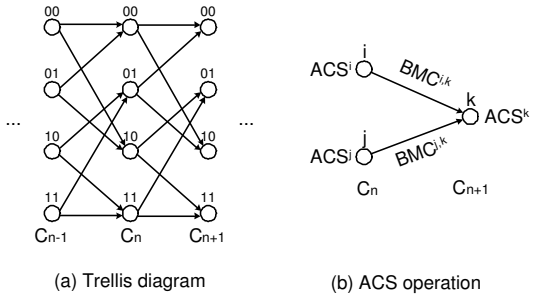

Fig. 2. (a) The trellis diagram of a channel encoder comprising 2 flip-flops; (b) The ACS operation where the source nodes i, j are in the n-th column and the destination node k is in the $(n+1)$-th column

The operation of the SOVA is divided into three steps: branch metric calculation (BMC), add compare select (ACS), and trace back (TB). All steps of the Viterbi algorithm are based on a trellis diagram that consists of nodes representing the state of the channel encoder and arrows representing the state transition of the channel encoder. Figure 2(a) is an example of a trellis diagram. From a computation perspective, the BMC operation is equivalent to calculating the distance between two points, and so it can be expressed as follows:

$$BMC^{i,j} = distance(r_{ij}, o_{ij}) = abs(r_{ij} - o_{ij}) \tag{1}$$

where i is the source node; j is the destination node; $r_{ij} \in \mathbf{I}$ is the received signal; and $o_{ij} \in \mathbf{I}$ is the error free output of a channel encoder corresponding to the state transition from node i to j. The BMC operation on all nodes in a trellis diagram can be done in parallel because the inputs of the BMC operation on a node are independent of the result of the BMC operation on other nodes.

Assuming there exist two input transitions at node k from nodes i and j, the ACS operation on a node k can be represented by following equation:

$$ACS^k = min(ACS^i + BMC^{i,k}, ACS^j + BMC^{j,k}) \tag{2}$$

From the above equation, we can see the operation dependency between ACS operations: ACS^k depends on ACS^i and ACS^j. Therefore, the ACS operations of all nodes can not be done in parallel. However, nodes i, j and k in the above equation additionally have the following relation:

$$\text{if } i, j \in C_n, \text{then } k \in C_{n+1} \tag{3}$$

where C_n is a set of nodes in the n-th column of a trellis diagram (see Figure 2(a)). In other words, the ACS operations of the nodes in the $(n+1)$-th

column can be done after the operations on the n-th column. Thus, at least the ACS operations of a column of a trellis diagram can be done in parallel.

The TB operation yields the most probable bit sequence which was transmitted by the transmitter based on the results for the BMC and ACS operation. Because the TB operation is similar to transversing a linked list, the operation is inherently sequential.

This whole set of steps are usually referred to as convolutional coding. If turbo coding is required, further operations are needed. The turbo decoder is based on the repeated application of 2 concatenated SOVA decoders. The output of each SOVA is interleaved, and then feed into the other SOVA. The number of iterations varies according to channel conditions. Under good conditions, early termination is possible. Because of the data dependency between the iterations, parallelization of the iterations of the turbo decoder is not possible.

The parallelism available on the channel decoders is related to the number of flip-flops used at the channel encoders because it determines the length of columns in a corresponding trellis diagram. In the W-CDMA physical layer, the convolutional encoder comprises 8 flip-flops, so the length of the corresponding column vector is $2^8 = 256$; and the turbo encoder uses 3 flip-flops so the length of the column vector is $2^3 = 8$. In addition, the number of columns in a trellis diagram is also determined by the number of flip-flops. It has been shown that $5(n+1)$ columns in a trellis diagram is sufficient for reliable decoding where n is the number of flip-flops in a channel encoder [10]. Therefore, the SOVA requires 45 $(= 5 \times (8+1))$ columns in the trellis diagram for the convolutional code, and 20 $(= 5 \times (3+1))$ columns for the turbo code. Because all BMC operations can be done in parallel, the maximum number of parallel BMC operation is 11520 $(= 256 \times 45)$ for the convolutional code and 160 $(= 8 \times 20)$ for the turbo code. Because the ACS operations of one column can be done in parallel, the maximum number of parallel ACS operation is 256 for the convolutional code and 8 for the turbo code. In order to increase the parallelism of the turbo decoder, it is possible to decode multiple trellis diagrams simultaneously. This is known as the sliding window technique.

Interleaver and Deinterleaver The interleaver and deinterleaver are used to overcome severe signal attenuation within a short time interval. In a wireless channel, an abrupt signal strength drop occurs very frequently. The interleaver in a transmitter randomizes the sequence of source information, and then the deinterleaver in a receiver recovers the original sequence by reordering. These operations scatter errors that occur within a short time interval over a longer time interval to reduce signal strength variation, and thus bit error rate, under the same channel conditions. Due to the randomness of the interleaving pattern, it is difficult to parallelize their operations without complex hardware support.

Modulator and Demodulator Modulation maps source information onto signal waveforms so that they carry source information over wireless links most efficiently. Demodulation extracts that information from the received signal. In

the W-CDMA physical layer, two classes of codes are deployed: channelization codes and scrambling codes.

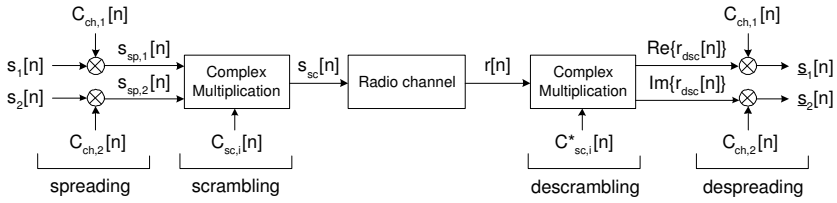

Fig. 3. Spreading/despreading and scrambling/descrambling operations with ignoring the operation of LPFs and analog frontend circuits

A channelization code is used for multiplexing multiple source streams into one physical channel. In the transmitter, the procedure to multiply a channelization code with a source sequence is called **spreading**, because it spreads out the energy of the source information over a wider frequency spectrum. The following equation shows the spreading operation:

$$s_{sp,1}[n] = \mathrm{C}_{ch,1}[n \ mod \ L_{ch}] \cdot s_1[\lfloor n/L_{ch} \rfloor] \tag{4}$$

$$s_{sp,2}[n] = \mathrm{C}_{ch,2}[n \ mod \ L_{ch}] \cdot s_2[\lfloor n/L_{ch} \rfloor] \tag{5}$$

where $s[n] \in \mathbf{B}$ is a source data sequence provided by the interleaver; $\mathrm{C}_{ch}[n] \in \mathbf{B}$ is a channelization code; L_{ch} is the length of the channelization code; *mod* is a modulo operation; and $\lfloor \cdot \rfloor$ is a floor function. **Despreading** at the receiver reconstructs the estimated source data sequences $\underline{s}_1[n]$ and $\underline{s}_2[n]$. This procedure is shown by the following equations:

$$\underline{s}_1[n] = \sum_{i=0}^{L_{ch}-1} \mathrm{C}_{ch,1}[i] \cdot Re\{r_{dsc}[n \cdot L_{ch} + i]\} \tag{6}$$

$$\underline{s}_2[n] = \sum_{i=0}^{L_{ch}-1} \mathrm{C}_{ch,2}[i] \cdot Im\{r_{dsc}[n \cdot L_{ch} + i]\} \tag{7}$$

where $r_{dsc}[n] \in \mathbf{I}^{\mathbf{C}}$ are the input of despreader which is provided by the descrambler. In the W-CDMA system, L_{ch} is a power of two from 4 to 512. It varies dynamically according to the source data rate such that the data rate of the output of spreading is fixed at 3.84 Mbps.

The use of a scrambling code enables us to extract the signal of one terminal when several terminals are transmitting signals at the same. The operation of multiplying a scrambling code with the output of the spreader is **scrambling** that is described by the following equation:

$$s_{sc}[n] = \mathrm{C}_{sc}[n \ mod \ L_{sc}] \cdot \{s_{sp,1}[n] + j s_{sp,2}[n]\} \tag{8}$$

where $\mathrm{C}_{sc}[n] \in \mathbf{B^C}$ is a scrambling code; L_{sc} is the length of the scrambling code; and $s_{sc,1}[n],\ s_{sc,2}[n] \in \mathbf{B}$ are the inputs of a scrambler that is generated by the spreaders. The corresponding action done in the receiver is **descrambling** which is described as follows:

$$r_{dsc}[n] = \mathrm{C}^*_{sc}[n \ mod \ L_{sc}] \cdot r[n] \tag{9}$$

where $r[n] \in \mathbf{I^C}$ is a received signal provided by low pass filters (LPF-Rx); and the * is a complex conjugate operation. In the W-CDMA physical layer, the data rate of all complex inputs and outputs of both scrambling and descrambling operations is fixed at 3.84 mega samples per second.

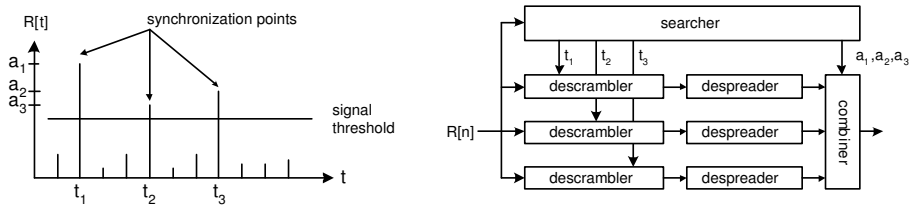

Fig. 4. (a) Example of cross correlation between the received signal and the scrambling code $C^*_{sc}[n]$ in a practical situation. Three synchronization points are detected by searcher at t_1, t_2, and t_3. (b) Structure of a rake receiver with three rake fingers. The operation times of each rake finger, t_1, t_2, and t_3, is set by searcher. The combiner aggregates the partial demodulation results of rake fingers.

One assumption on the operation of the descrambler and despreader is that the receiver is perfectly synchronized with a transmitter. To achieve time synchronization, a receiver computes the cross correlations between the delayed version of a received signal $r[n-\tau] \in \mathbf{I^C}$, and a conjugated scrambling code sequence $\mathrm{C}^*_{sc}[n] \in \mathbf{B^C}$, by varying τ as follows:

$$R[\tau] = \sum_{i=0}^{L_{cor}-1} \mathrm{C}^*_{sc}[i] \cdot r[i+\tau], \text{ where } 0 \le \tau \le (L_s - 1) \tag{10}$$

where L_{cor} is the correlation length. In an ideal situation, $R[\tau]$ is maximized at the synchronization point because of the auto correlation property of the scrambling code: $\frac{1}{N}\sum_{i=0}^{N-1} C_{sc}[i] \cdot C^*_{sc}[i-\tau] = \delta[\tau]$. However, in a practical situation there are several correlation peaks because of multipath fading that an identical radio signal term arrives at a receiver multiple times with random attenuation and propagation delay. The multipath fading is caused by the reflection of the radio signal from objects placed on the signal propagation path. The **searcher** is an entity that finds synchronization points where the cross correlation $R[\tau]$ is greater than a predefined threshold level. Specifically, the operation of the

searcher can be divided into four steps: 1) $R[\tau]$ calculation; 2) detection of local correlation peaks by calculating the derivative of $R[\tau]$; 3) filtering out high frequency noise terms from the local correlation peaks; and 4) global peaks detection by sorting the filtered local correlation peaks. The steps (1) and (2) have a high level of parallelism, whereas the steps (3) and (4) are difficult to parallelize. Details of the searcher can be found in [11]. The L_{cor} and L_s are important design parameters that significantly affect the workload of the W-CDMA physical layer. In our implementation, the L_{cor} and L_s are assumed to be 320 and 5120 correspondingly.

The receiver of the W-CDMA system descrambles and despreads a received signal at each of the synchronization points which are detected by the searcher, and then the partial demodulation results of the synchronization points are aggregated. This generation of partial demodulation results with proper delay compensation and the aggregation of these partial demodulation results mitigates the effect of the multipath fading. The aggregation of partial demodulation results is performed by a **combiner**. A receiver structure comprising the searcher, multiple demodulation paths, and the combiner is called **rake receiver** [12]. One demodulation path consisting of a descrambler and despreader is called as **rake finger**. The rake receiver is the most popular architecture used for CDMA terminals. In our implementation, we assumed the maximum number of rake fingers to be 12.

Low Pass Filter A low pass filter (LPF) filters out signal terms that exist outside of an allowed frequency band in order to reduce the interference. Filtering can be represented by an inner product between an input data vector and a coefficient vector as follows:

$$y[n] = \sum_{i=0}^{L_{LPF}-1} c_i \cdot x[n-i] \tag{11}$$

where $x[n]$ is the input sequence to be filtered; the $c_i \in \mathbf{I}$ are the filter coefficients; and L_{LPF} is the number of filter coefficients.

There are two kinds of LPFs in the W-CDMA physical layer: LPF-Tx and LPF-Rx. Although the functionality of both filters are identical, the workload of both filters are different. The first difference is the size of operand. $x[n] \in \mathbf{B}$ in the LPF-Tx, but $x[n] \in \mathbf{I}$ in the LPF-Rx. The second difference is the number of LPF entities. At the LPF-Rx, there are two LPFs: one for the real part of the signal and the other for the imaginary part. However, the LPF-Tx consists of six LPFs (refer to the LPF-Rx in Figure 5). This structure is the result of an effort to reduce the amount of multiplication. A detailed explanation on the LPFs of the W-CDMA system can be found in [13]. In our implementation, L_{LPF} is 65.

Power Control In general, CDMA systems adaptively control transmission power so that signals can be sent over the wireless channel at a minimum power level while satisfying a target bit error rate. The random variation of

radio channel characteristics requires a feedback loop to control the strength of a transmitted signal. A transmitter sends reference signals called pilots, then a receiver sends back power control commands according to the quality of received pilot signals. To evaluate signal quality, it is necessary to fully demodulate the pilot signal in realtime. This sets a hard realtime requirement in the LPF, spreader/despreader, scrambler/descrambler and combiner. In the W-CDMA physical layer, the frequency of the power control operation is about 1.5KHz – every 0.67 msec.

Operation State From the view point of processor activity, it is possible to divide the operation of a W-CDMA terminal into three states[4]: idle, control hold, and active state. In the idle state a wireless terminal does not provide any application service to the user. However, even in this state, a terminal must be ready to respond to control commands from basestations. Although only simple tasks are performed in the idle state, the power consumed in this state is significant because a terminal spends most of its time in this state. In the active state, a terminal transmits and receives user data. It is the most heavily loaded operation state, because all function blocks are active. The control hold state is defined to represent the operation of a terminal during short idle periods between packet bursts. In the control hold state, a terminal maintains a low bandwidth control connection with basestations for fast transition to the active state when packets arrive.

3 Workload Analysis

Table 1. Assumed operation conditions of a W-CDMA terminal for workload analysis

Representation service	Service type	Packet service
	Data link	2 Mbps / 128 Kbps
	Signaling link	3.4 Kbps bidirectional
Channel condition	# of basestations	3
	# of rake fingers	12
	# of average turbo iterations	3

Terminal Operation Conditions Before going into the detailed workload analysis, we need to clarify the operation conditions of the W-CDMA terminal,

[4] In the W-CDMA standard, there are five radio resource control (RRC) states; camping on a UTRAN cell, ura_PCH, ura_FACH, cell_FACH, and cell_DCH states [14]. These RRC states are defined from the perspective of radio resource management. We redefine the operation state of the W-CDMA terminal according to processor activity. In our definition, the camping on a UTRAN cell, ura_PCH, ura_FACH, and cell_FACH states are grouped into the idle state. The cell_DCH state is divided into the control hold and active state.

because the workload of the W-CDMA physical layer is affected by several things: 1) operation state; 2) application type; and 3) radio channel status. Because the operation state significantly affects the hardware for the W-CDMA physical layer, we will analyze the variation in workload for all operation states. However, we limit the application type and radio channel condition.

Generally, we can classify application services into two categories: circuit service and packet service. Circuit service is a constant data rate service such as a voice call. Packet service is a variable data rate service such as internet access. Because the packet arrival pattern of the packet service demands a more complex resource management scheme, we select the packet service as our representative service. In addition, we further assume an asymmetric packet service that consists of a 2 Mbps link in the direction from basestation to terminal and a 128 Kbps link in the reverse direction. The asymmetric channel assumption matches the behavior of most packet services, for instance web browsing. For control signaling, the packet service additionally has a bidirectional signaling link with a 3.4 Kbps data rate.

The workload of a W-CDMA terminal is also varied by three radio channel conditions: 1) the number of basestations that communicate with a terminal at the same time; 2) the number of correlation peaks resulted in by the multipath fading; and 3) the quality of received signal. We assume 3 basestations, and 4 correlation peaks from the signal of a basestation. Thus, the W-CDMA terminal activates a total of 12 ($= 3 \times 4$) rake fingers. Because he quality of the received signal has a direct impact on the number of iterations of the turbo decoder, we assumed the quality of the received signal is set by the power control such that the average number of turbo decoder iterations is 3 per frame.

System Block Diagram Figure 5 shows a detailed block diagram of the W-CDMA physical layer. It explains which algorithms participate in the action of each operation state. In the idle state, a subset of the reception path, the LPF-Rx and rake receiver, is active. In the control hold state, a bidirectional 3.4 Kbps signaling link is established with the basestations. Thus, a terminal activates both transmission and reception paths including the convolutional encoder/Viterbi decoder, LPF-Rx/Tx, modulator/demoulator, and power control. In the active state, a terminal additionally establishes a bidirectional high speed data link as shown in Table 1. Thus, the turbo encoder/decoder participate in the active state operation of a terminal.

Figure 5 also describes the interface between the algorithms of that make up W-CDMA. The number at the top of each arrow represents the number of samples per second, and that at the bottom represents the size of a sample. From these numbers, we can derive the amount of traffic between the algorithms. The size of most data in the transmission path is 1 bit, but it is 8 or 16 bit in the reception path because the channel decoders use soft numbers as explained previously. From the diagram, we can see that the data rate is abruptly changed by the spreader and despreader. In the transmission path, the data rate is up-

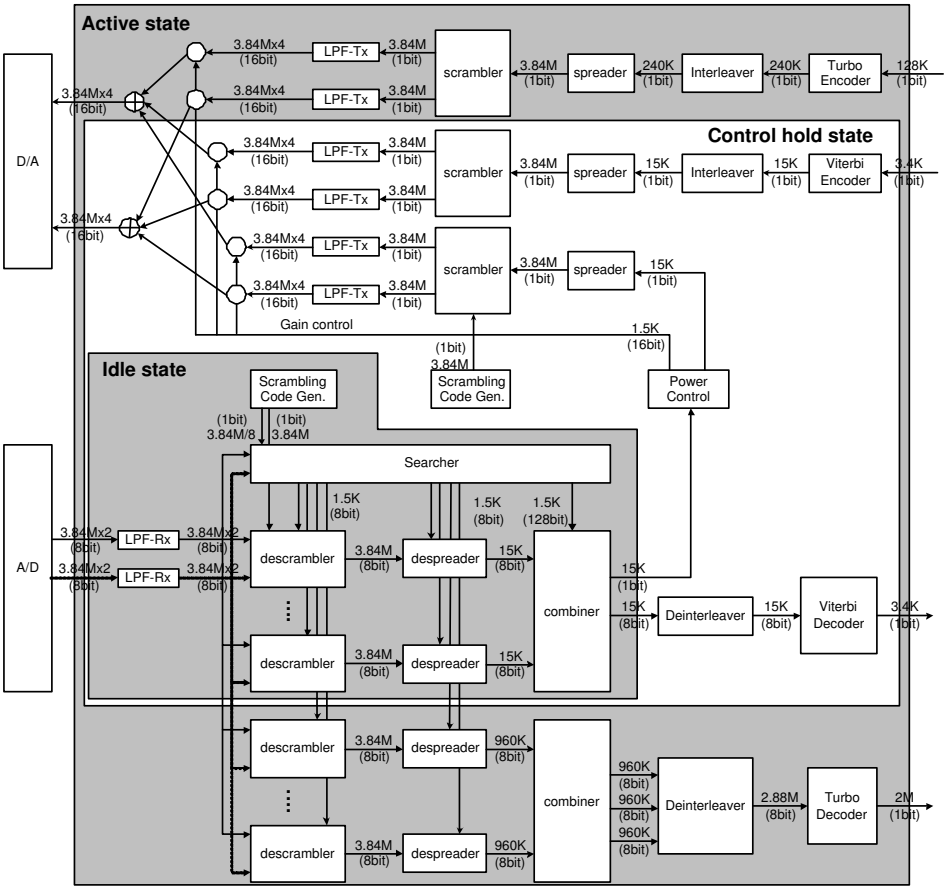

Fig. 5. Detailed block diagram of the W-CDMA physical layer providing the packet service described in Table 1.

converted from kilo sample per second to mega samples per seconds after the spreading operation. The reception path exhibits the reverse.

Table 2. Processing time requirements of the W-CDMA physical layer

<table>
<tr><th></th><th colspan="2">Active/Control hold state</th><th>Idle state</th></tr>
<tr><th></th><th>Processing Time(ms)</th><th>Execution Freq.(Hz)</th><th>Processing Time(ms)</th></tr>
<tr><td>Searcher</td><td>Fixed(5*)</td><td>50*</td><td>Fixed(40*)</td></tr>
<tr><td>Interleaver/Deinterleaver</td><td>Fixed(10)</td><td>100</td><td>-</td></tr>
<tr><td>Convolutional encoder</td><td rowspan="3">Fixed(10/20/40)</td><td rowspan="4">Variable</td><td rowspan="4">-</td></tr>
<tr><td>Viterbi decoder</td></tr>
<tr><td>Turbo encoder</td></tr>
<tr><td>Turbo decoder</td><td>Variable(10∼50*)</td></tr>
<tr><td>Scrambler/Descrambler</td><td rowspan="5">Fixed(0.67)</td><td rowspan="5">1500</td><td rowspan="3">Fixed(0.67)</td></tr>
<tr><td>Spreader/Despreader</td></tr>
<tr><td>LPF-Rx</td></tr>
<tr><td>LPF-Tx</td><td>-</td></tr>
<tr><td>Power control</td><td></td></tr>
</table>

Processing Time Table 2 shows that the W-CDMA physical layer is a mixture of algorithms with various processing time requirements. The * notation in the table indicates that the corresponding parameter is a design parameter. Other parameters in the table are explicitly specified by the W-CDMA standard. The processing time shown in the second and fourth columns is the allocated time for one call to each algorithm. The task frequency in the third column is the number of times each algorithm is called within a second.

We assume that the searcher is executed every 20 msec because a radio channel can be considered as unchanging during this interval. The scrambler, spreader, and LPF have periodic and very strict processing time requirements, because they participate in the power control action. The convolutional code is mainly used for the circuit service with a constant data rate, so the Viterbi decoder needs to fulfill its operation before the arrival of the next frame to avoid buffer overflow. The processing time of the Viterbi decoder can be configured with 10, 20, or 40 msec intervals according to how the service frame length is configured. Whereas the turbo code aims the packet service with a burst packet arrival pattern. By buffering of bursty packets, can relax its processing time constraint significantly. We assume that the processing time of the turbo decoder varies between 10∼50 msec according to the amount of buffered traffic. In the idle state, tasks have loose timing constraints, so the searcher operation is sequentially performed with minimal hardware and the task frequency is not a concern.

Table 3. Peak workload profile of the W-CDMA physical layer and its variation according to the operation state

	Active		**Control Hold**		**Idle**	
	(MOPS)	%	**(MOPS)**	%	**(MOPS)**	%
Searcher	**26538.0**	**42.1**	**26358.0**	**58.4**	**3317.3**	**37.7**
Interleaver	2.2	0.0	2.2	0.0	-	-
Deinterleaver	0.2	0.0	0.2	0.0	-	-
Conv. encoder	0.0	0.0	0.0	0.0	-	-
Viterbi Decoder	200.0	0.3	200.0	0.4	-	-
Turbo encoder	0.0	0.0	0.0	0.0	-	-
Turbo decoder	**17500.0**	**27.8**	**0.0**	**0.0**	-	-
Scrambler	245.3	0.4	245.3	0.5	-	-
Descrambler	**2621.4**	**4.2**	**2621.4**	**5.8**	**889.2**	**10.1**
Spreader	297.5	0.5	297.5	0.7	-	0.0
Despreader	**3642.5**	**5.8**	**3642.5**	**8.0**	**607.1**	**6.9**
LPF-Rx	**3993.6**	**6.3**	**3993.6**	**8.8**	**3993.6**	**45.3**
LPF-Tx	**7897.2**	**12.6**	**7897.2**	**17.4**	-	-
Power control	0.0	0.0	0.0	0.0	-	-
Total	62937.0	-	45272.9	-	8807.2	-

Workload Profile The detailed workload profile of the W-CDMA physical layer is shown in Table 3. For this analysis, we compiled our W-CDMA benchmark with an Alpha gcc compiler, and executed it on the M5 architectural simulator [15]. We measured the instruction count that is required to finish each algorithm. Peak workload of each algorithm is achieved by dividing the instruction count by the tightest processing time requirement of each algorithm shown in Table 2.

The first thing to note in Table 3 is that the total workload varies according to the operation state change. The total workloads in the control hold and idle states are about 72% and 14% of that in the active state. Second, the workload profile also varies according to the operation state. In the active and control hold states, the searcher, turbo decoder, LPF-Tx are dominant. In the idle state, the searcher and LPF-Rx are dominant.

Intrinsic Computations Major intrinsic operations in the W-CDMA physical layer operation is listed in Table 4. As we discussed in Section 2, many algorithms in the W-CDMA physical layer are based on multiplication operations. Because multiplication is a power consuming operation, it is advantageous to simplify this into operations. First, the multiplications in the spreader and scrambler can be simplified to an exclusive OR, because both operands are either 1 or -1. By mapping $\{1, -1\}$ to $\{0, 1\}$, we can use the exclusive OR operation instead of multiplication. Second, the multiplications in the searcher, descrambler, despreader, and LPF-Tx can be simplified into conditional complement operations, because one operand of the multiplications in these algorithms is either -1 or 1, and

Table 4. Intrinsic computations in the W-CDMA physical layer

Operations	**Algorithms**	**Description**
Exclusive OR	Spreader, Scrambler	$z = x \oplus y$
Conditional Complement	Searcher, Descrambler Despreader, LPF-Tx	$z = s\ ?\ x : -x$
Multiplication	LPF-Rx	$z = c \cdot x$
Scalar reduction	Searcher, Despreader, LPF	$z = \sum_{i=0}^{N-1} x_i$
Vector permutation	Viterbi/Turbo decoder, Searcher	$z[n] = x[n + p_n]$
BMC	Viterbi/Turbo decoder	$z = abs(x_0 - x_1)$
ACS	Viterbi/Turbo decoder	$z = min(x_0 + c_0, x_1 + c_1)$

the other operand is a fixed point number. However, the multiplication of the LPF-Rx cannot be simplified because both operands are fixed point numbers. The operands of the multiplications in Equations (8), (9), and (10) are complex numbers. We can treat these operations as integer multiplications, because of the following relation: $(a + jb)(c + jd) = (ac - bd) + j(bd + ad)$.

For the frequent inner product operations of the searcher, despreader, and LPFs, we need a scalar reduction operation to add up all elements in a vector. A vector permutation is also required for the channel decoders and searcher, because either output or operand vector needs to be permuted. In channel decoders, the core operations are the BMC and ACS.

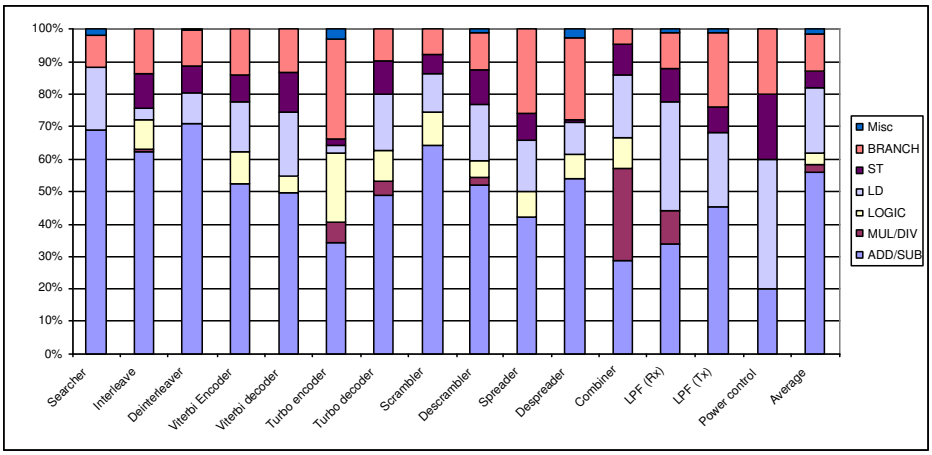

Fig. 6. Instruction type breakdown result

Instruction Type Breakdown Figure 6 shows the instruction type breakdown for the W-CDMA physical layer and their weighted average. To obtain these results, a terminal is in the active state with peak workload. Instructions are grouped into seven categories: add/sub; multiply/divide; logic; load; store; branches; and miscellaneous instructions. Because all algorithms are implemented with fixed point operations, there are no floating point instruction types shown here.

The first thing to notice is the high percentage of add/subtract instructions. They account for almost 50% of all instructions, with the searcher at about 70%. This is because the searcher's core operation is the inner product of two vectors, and the multiplication in an inner product can be simplified into a conditional complement. Furthermore, the complement operation is equivalent to a subtraction. Other hotspots, like the turbo decoder, also have a large number of addition operations. In the BMC and ACS of the turbo decoder, the additions are for calculating the distance between two points. In the TB of the turbo decoder, the additions come from pointer chasing address calculation.

The second thing to notice is the lack of multiplications/divisions ($\sim$3% on average). This is because the multiplications of major algorithms are simplified into logical or arithmetic operations as discussed earlier. The multiplication of the combiner and turbo encoder is not a significant because their workload is very small as shown in Table 3. One exception is multiplications in the LPF-Rx. Figure 6 also shows that the number of load/store operations are significant. This is because most algorithms consist of loading two operands and storing the operation result.

Results also show that the portion of branch operations is about 10% on average. Most frequent branch patterns are loops with a fixed number of iterations corresponding to a vector size, and a conditional operation on vector variables. There are a few while or do-while loops, and most loops are 1 or 2 levels deep.

Parallelism To meet the W-CDMA performance requirements in software, we must exploit the inherent algorithmic parallelism. Table 5 shows a breakdown of the available parallelism in the W-CDMA physical layer. We define data level parallelism (DLP) as the maximum single instruction multiple data (SIMD) vector width and thread level parallelism (TLP) as the maximum number of different SIMD threads that can be executed in parallel. The second and third columns in the table are the ratio between the run time of the scalar code and the vector code. The fourth column represents maximum possible DLP. Because a vector operation needs two operands, we separately represent the bit width of two vector operands in the fifth column. The last column shows the TLP information.

From Table 5, we can see that the searcher, LPF, scrambler, descrambler, and the BMC of the Viterbi decoder contain large amounts of the DLP and TLP. For the case of the scrambler and descrambler, it is possible to convert the DLP into TLP by subdividing large vectors into smaller ones. Although it is one of dominant workloads, the turbo decoder contains limited vector parallelism

Table 5. Parallelism available in the algorithms of the W-CDMA physical layer

		Scalar workload (%)	Vector workload (%)	Vector width (elements)	Vector element width (bit)	Max concurrent Thread
Searcher		3	97	320	1,8	5120
Interleaver		100	0	-	-	-
Deinterleaver		100	0	-	-	-
Viterbi encoder		60	40	8	1,1	1
Viterbi Decoder	BMC	1	99	256	8,8	45
	ACS	1	99	256	8,8	45
	TB	100	0	-	-	-
Turbo encoder		60	40	4	1,1	2
Turbo Decoder	BMC	1	99	16	8,8	20
	ACS	1	99	16	8,8	20
	TB	100	0	-	-	-
Scrambler		1	99	2560	1,1	1
Descrambler		1	99	2560	1,8	1
Spreader		100	0	-	-	-
Despreader		100	0	-	-	-
Combiner		100	0	-	-	-
LPF-Tx		1	99	65	1,16	6
LPF-Tx		1	99	65	8,8	2
Power Control		100	0	-	-	-

because the allowed maximum vector length of the ACS operation of the turbo decoder is 8.

There are also many unparallelizable algorithms in the W-CDMA physical layer. The interleaver, deinterleaver, spreader, despreader, and combiner operations have little DLP and TLP. Fortunately, the workload of these algorithms is not significant as show in Table 3. Therefore we can easily increase system throughput by exploiting the inherent parallelism shown in Table 5.

Memory Requirement Because memory is one of the dominant power consuming elements in most systems, the analysis of the characteristics of memory is important. In general, there are two types of memory in a hardware system: data memory and instruction memory. Table 6 presents the data and instruction memory for all the algorithms in W-CDMA.

Columns 2∼7 in Table 6 show the size of the required data memory. The data memory is further divided into two categories: I/O buffer and scratch pad. The I/O buffer memory is used for buffering streams between algorithms. The scratch pad memory is temporary space needed for algorithm execution. We analyzed both size and access bandwidth of the data memory.

Table 6. Memory requirements of the algorithms of the W-CDMA physical layer

	Data memory (Kbyte)						**Inst. Memory** (Kbyte)
	I-buffer		**O-buffer**		**Scratch pad**		
	Kbyte	Mbps	Kbyte	Mbps	Kbyte	Mbps	
Searcher	**20.8**	**2.1**	**0.1**	**0.1**	**32.0**	**2654.3**	**3.1**
Interleaver	1.2	1.1	1.2	1.1	9.5	1.9	0.1
Deinterleaver	**26.1**	**5.2**	**26.1**	**5.2**	**8.7**	**5.2**	**0.1**
Viterbi Encoder	0.1	0.1	0.1	0.1	0.1	0.1	0.3
Viterbi Decoder	0.1	0.1	0.1	0.1	4.8	2.1	1.6
Turbo Encoder	2.6	4.0	7.8	12.0	0.1	2.0	1.6
Turbo Decoder	**61.5**	**96.0**	**2.6**	**4.0**	**6.4**	**25600.0**	**3.4**
Scrambler	0.7	15.4	0.7	15.4	0.7	15.4	0.5
Descrambler	**5.6**	**123.2**	**5.6**	**123.2**	**0.7**	**15.4**	**0.5**
Spreader	0.1	1.9	0.4	7.6	0.1	3.9	0.4
Despreader	0.4	7.6	0.1	1.9	0.1	3.9	0.3
Combiner	0.1	0.1	0.1	0.1	0.1	0.1	0.1
LPF-Tx	**0.3**	**7.6**	**10.3**	**245.8**	**0.1**	**1996.8**	**0.2**
LPF-Rx	**10.5**	**245.8**	**2.5**	**61.4**	**0.1**	**1996.8**	**0.2**
Power control	0.1	0.1	0.1	0.1	0.1	0.1	0.1

From the table, we can see that the W-CDMA algorithms require a small amount data memory, generally less than 64 Kbyte. In addition, we can see that the scratch pad memory is the most frequently accessed, especially in the searcher, turbo decoder, and LPFs. The access of I/O memory does not occupy a significant portion at the total memory access.

The last column of Table 6 shows the instruction memory size for each algorithm. For the analysis of the instruction memory size, we compiled our benchmark program on an Alpha processor. The average code size is less than 1 Kbyte and most kernels are below 0.5 Kbyte. This result is typical of many digital signal processing algorithms.

4 Architectural Implications

System Budget The power budget allocated to baseband signal processing in commercial wireless terminals is typically between 100∼300 mW. Using this power budget and the total peak workload data from Table 3, we calculated that the power efficiency of a processor for the W-CDMA physical layer must be between 210∼630 MOPS/mW. Because W-CDMA is one of the leading emerging wireless communication networks, we take this result as the system budget for typical SDR applications. In addition, we have calculated the maximum performance and energy efficiency of three conventional architectures: a digital signal processor (DSP), an embedded processor, and a general purpose processor (GPP). As shown in Table 7, these conventional architectures are a long way off of satisfying the requirements of SDR.

Table 7. Estimation of system budgets for SDR applications and the achievable level of some typical commercial processors

	Power Efficiency (MOPS/mW)
Budget for SDR applications	210∼630
DSP: TI TMS320C55X [16][17]	∼40
Embedded Processor: AD ADSP-TS201 Tigersharc [18][19]	∼5
GPP: AMD Athelon MP [20][21]	∼0.003

SIMD Processor One key aspect for supporting the W-CDMA physical layer is to use an SIMD architecture. There are two main reasons why an SIMD architecture is beneficial. First, is that the W-CDMA physical layer contains a large amount of DLP as shown in Table 5. Second, the types of computations in tasks with high levels of DLP are mainly add/subtract operations. The absence of complex operations allows us to duplicate the SIMD functional units with minimal area and power overhead. Furthermore, we can expect significant power gain because a SIMD architecture can execute multiple data elements with one instruction decoding.

The intrinsic instructions shown in Table 4 can be used as a guide to designing the datapath and instruction set of the SIMD processor. In addition to the narrow data widths and simple operations, the datapath of the SIMD processor needs a vector status register that stores the status for each element of vector operations such as carry, zero and overflow. These vector status registers are useful for the realization of the conditional complement shown in Table 4. The datapath of the SIMD processor also needs a shuffle network to support vector permutation operations. The output of the arithmetic unit is shuffled before being stored in the register file. The hardware complexity of the shuffle network is exponentially proportional to the number of input nodes, which sets a limit on the width of the SIMD processor.

Scalar Processor In addition to SIMD support, the W-CDMA physical layer requires scalar support because there are many small scalar algorithms in the W-CDMA physical layer as shown in Table 5. These scalar algorithms can be broken down into two main types: control and management tasks like the power control and rake receiver; and computation intensive DSP operations like the TB of the turbo decoder. Most of the control operations have tight response time requirements, so there is a need for realtime interrupt support in the scalar processor. A conventional low power GPP such as an ARM processor is adequate for the scalar processor.

Multiprocessor To further enhance the performance, a multiprocessor architecture is desirable if the TLP in the W-CDMA physical layer is to be exploited. Implementing a multiprocessor architecture raises many design questions: the

granularity of the processing element (PE); the application task partitioning and mapping; the inter-process communication (IPC) mechanisms; and the heterogeneity of the PE.

One key factor which determines the efficiency of a multiprocessor architecture is the amount of IPC. Communication over an IPC network takes longer and dissipates more power than an internal memory structure. Because of this, the size of the PEs and the method of mapping and partitioning application tasks should be determined so as to minimize IPC. In addition, the choice of the IPC mechanism is important, because the communication pattern of the W-CDMA physical layer is point to point. This favors message passing, because copying of data from one PE to another is more power efficient.

Another method for enhancing performance is using heterogeneous PEs. It is possible to save chip area by implementing the multiplier only on a subset of PEs because multiplication is not used for all W-CDMA algorithms. Though heterogeneous PEs are desirable, homogeneous PE reduces development cost by reusing PE design.

Memory Hierarchy In the memory hierarchy, certain choices optimizing for power is important. In the local memory of the PEs, the size of the memory is crucial for power efficiency, because, as shown in Table 6, there is very high internal memory traffic in the algorithms. Minimizing the size of these memories is necessary for more power efficient accesses. Also a global memory should be used because the W-CDMA physical layer requires the buffering of large bursty data traffic–input data traffic to the turbo decoder. This removes the need for large and power inefficient local memories.

For the W-CDMA physical layer, data and instruction caches are not essential. The memory access pattern of the W-CDMA physical layer is highly deterministic and exhibits very dense spacial locality, so it is possible to schedule memory access patterns in small sized memories. The advantages of cache free memories are low power consumption and a deterministic operation time. In DSP applications, a deterministic operation time is necessary for meeting timing requirements and easy system validation.

Power Management Two power management challenges arise from the workload profile presented in Table 3: (1) Optimize the system for wide workload variations in the active and control hold states; (2) Minimize the power consumption in the idle state. Dynamic voltage scaling (DVS) and dynamic frequency scaling (DFS) are attractive ways to tackle the first challenge. The operation voltage and frequency can be dynamically adjusted according to the variation of wireless channel conditions and user traffic. In order to minimize the power consumption in the idle state, we can selectively turn-off blocks which are not required in this state, such as the LPF-Tx and turbo decoder. In addition, the operation of the LPF-Rx and search must be optimized for power, because these tasks are the leading power consumers in the idle state.

5 Conclusion

Software defined radio presents a new challenge for the architecture community. A programmable SDR solution needs to offer supercomputer performance, while running on mobile devices with ultra low power consumption. In this paper, we have presented a complete W-CDMA system benchmark and an analysis of its computational characteristics. The benchmark includes realistic operating conditions, and algorithm configurations that are based on commercial W-CDMA implementations. From our study, W-CDMA shows very unique dynamic behavior characteristics. It has ultra high performance requirements, and dynamically changing real-time processing requirements. The algorithms are dominated by addition/subtraction, small memory footprints, and a high degree of data and task level parallelism. The results indicate that a programmable architecture needs SIMD as well as scalar support, can be optimized for narrow width operations, requires only small instruction and data memories, and can exploit dynamic voltage scaling to account for dynamic workload changes. Our benchmarks provide a useful evaluation for future architecture studies of SDR. In the future, we will include 802.11x protocols into our benchmark suite to gain further understanding of the programmability and computation requirements of wireless protocols.

References

1. Austin, T., et al.: Mobile Supercomputers. IEEE Computer **37** (2004) 82-84
2. Tuttlebee, W.: Software Defined Radio: Baseband Technology for 3G Handset and Basestations. 1st edn. John Wiley & Sons, New York (2002)
3. Holma, H., Toskala, A.: W-CDMA for UMTS: Radio Access for Third Generation Mobile Communications. John Wiley & Sons, New York (2000)
4. http://www.sandbridgetech.com/
5. http://gram.eng.uci.edu/morphosys/
6. http://www.3p1t.com/
7. Forney, G., D., Jr.: The Viterbi Algorithm. Proc. IEEE **61** (1973) 268-278
8. Berrou, C., Glavieux, A., Thitimjshima, P.: Near Shannon Limit Error Correcting Coding and Decoding: Turbo-Codes. ICC (1993)
9. 3GPP TS 25.211, Multiplexing and Channel Coding (FDD)
10. Parhi, K., Nishitani, T.: Digital Signal Processing for Multimedia Systems. 1st edn. Marcel Dekker, New York (1999)
11. Grayver, E., et al.: Design and VLSI Implementation for a WCDMA Multipath Searcher. IEEE Trans. on Vehicular Technology **54** (2005) 889-902
12. Rappaport, T.: Wireless Communications: Principles and Practice. IEEE Press, Piscataway (1996)
13. Berenguer, I., et al.: Efficient VLSI Design of a Pulse Shaping Filter and DAC interface for W-CDMA transmission. 16th IEEE International ASIC/SoC Conference (2003)
14. 3GPP TS 25.331: Radio Resource Control (RRC) Protocol Specification
15. Binkert, N., Hallnor, E., and Reinhardt, S.: Network-Oriented Full-System Simulation using M5. CAECW (2003)

16. http://www.bdti.com/procsum/tic55xx.htm
17. Dielissen, J., et al.: Power-Efficient Layered Turbo Decoder processor. DATE (2001)
18. http://www.analog.com/
19. Bursky, D.: DSPs Attack Throughput Needs with 600-MHz Clocks and eDRAM. Electronic Design **51** (2003)
20. http://www.amd.com/
21. Feng. W.: Honey, I Shrunk the Beowulf!. ICPP (2002)