A Case Study of Mapping a Software-Defined Radio (SDR) Application on a Reconfigurable DSP Core

Behzad Mohebbi, Eliseu Chavez Filho,
Rafael Maestre, Mark Davies
Morpho Technologies Inc
19772 Mac Arthur Blvd, Irvine, CA 92612
{behzad, ecf, rafael, mdavies}@morphotech.com

Fadi J. Kurdahi
EECS Department,
University of California
Irvine, CA 92697
kurdahi@uci.edu

ABSTRACT

We present a case study involving the implementation of a complete Wideband CDMA (WCDMA) digital receiver part of an AMR channel onto a reconfigurable core. WCDMA is one of the two major standards for the third generation (3G) cellular systems. Traditionally most of the receiver components were confined to ASIC implementation for performance, size and power consumption reasons. The MS1 reconfigurable DSP core provides both a microprocessor and reconfigurable fabric as well as a variety of peripherals. The various functions of the receiver were mapped onto different core components. The complete system was tested both in simulation as well as on a hardware platform comprising a silicon implementation of the MS1 DSP core.

Categories: C3 Real-time Embedded systems

General Terms: Algorithms, Design, Experimentation

Keywords: Software-Defined Radio, Reconfigurable Computing

1. Introduction

Reconfigurable computing systems represent a hybrid approach between the paradigms of general-purpose and application-specific systems. They combine a software programmable processor and a reconfigurable hardware component that can be reused in different applications. This combination allows reconfigurable systems to achieve performance levels significantly higher than obtained with general-purpose systems and with wider flexibility than that offered by application-specific systems. This paper introduces the Morpho Technologies' MS1 reconfigurable DSP (rDSP) Core. The MS1 rDSP Core is primarily targeted for applications with inherent parallelism, word-level granularity and computation-intensive characteristics. Examples of key applications are baseband processing in digital wireless communication systems, and video compression and image processing in multimedia systems.

One of the key components in the case of mobile or airborne devices is the Software-Defined Radio (SDR). SDR promises to revolutionize the way such systems are implemented. SDR is the implementation of wireless network protocol whose operation modes can be changed after manufacturing. The relegation of more functions to programmable DSP or DSP-like hardware enables high performance, ubiquitous wireless terminals, infrastructure, equipment and services that allow seamlessness across diverse networks and integration of capabilities that encompass multiple standards and operational paradigms.

One of the key target standards that can benefit from an SDR approach is the third generation wireless system. Third-generation (3G) wireless standards such as WCDMA or CDMA2000 present computationally demanding tasks that until now have been dealt with only by application-specific cores. These ASICs typically perform chip-level baseband processing (e.g. the RAKE receiver) as well as encoding/decoding of source

information (e.g. voice and video). Conventional DSPs are usually employed for Forward Error Correction except in the case of Turbo decoding, which is implemented in ASIC logic due to its performance requirements. The Medium-Access Control (MAC) layer typically runs on an off-the-shelf microcontroller core.

ASICs present several disadvantages in this context: they have to be designed for the worst case processing scenario; they increase the time-to-market due to the long design cycle time; they increase cost as several ASIC cores have to be utilized in such multi-mode application; and finally they can not adapt to new standard releases, as it is impossible to do modifications after silicon tapeout.

We have proven in practice that a single MS1 rDSP Core is capable of performing all the chip-level baseband processing as well as source encoding/decoding (both voice and video) without the need for ASIC blocks. The results are simplification of design, cost reduction due to the simpler design, faster time to market, and smaller silicon area and flexibility for modifications after product delivery.

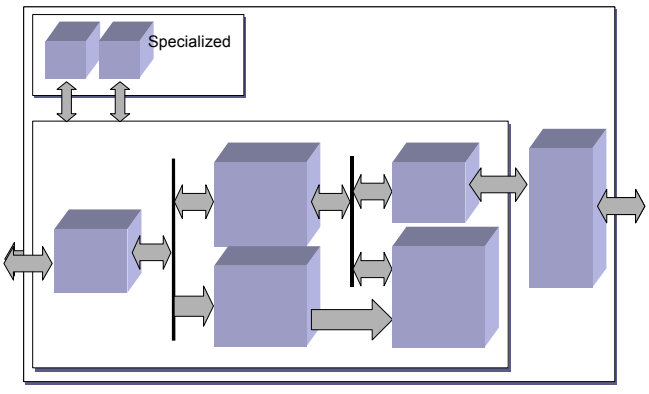

Figure 1: The MS1 Core Architecture

CODES+ISSS'03, October 1–3, 2003, Newport Beach, California, USA.

2. The MS1 rDSP Architecture

Morpho Technologies provides Reconfigurable DSP (rDSP) solutions based on a reconfigurable array processing paradigm, known as MS1 rDSP, which is briefly discussed here.

As in conventional reconfigurable systems [1], the MS1 Core contains a reconfigurable block, called the RC Array, and a 32-bit RISC processor, called the mRISC (Figure 1). The RC Array consists of Reconfigurable Cells (RCs) interconnected by a reconfigurable network. Both the functionality of the RCs and the network interconnections are determined by a configuration program, called Context. By writing the appropriate context, the developer can use the RC Array to exploit the parallelism available in the application. The mRISC Core Controller determines the application's control flow, executes the sequential tasks of the application, and starts transfers to/from the off-core memory. The instructions for the mRISC are stored in the Instruction Memory.

The MS1 Core also contains a Context Memory, a Data Buffer and an I/O Controller. The Context Memory stores the context words for the RC Array. The context words configure the operations performed by the RCs and their interconnectivity. The Data Buffer stores the input data to the RC Array as well as the produced results. The Data Buffer allows data input/output to be overlapped with data processing. The I/O Controller performs context loading from the off-core memory into the Context Memory and data transfers between the Data Buffer and the off-core memory. The I/O Controller receives transfer requests from the mRISC and generates an interrupt once the transfer is complete.

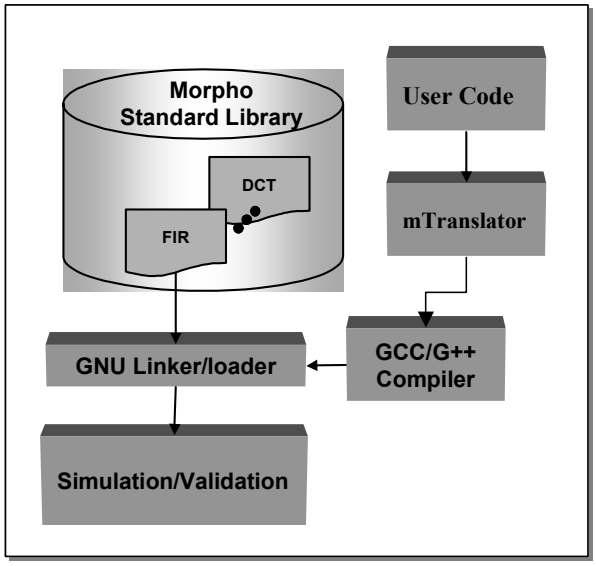

Figure 2: MS1 Programming Model

The Reconfigurable Cell is the programmable element in the RC Array. It performs general-purpose operations as well as word- and bit-level DSP functions. Input operands can be either internal to the RC, or from other RCs or from the Data Buffer. Due to the flexible interconnect structure and reconfiguration capabilities, the RC Array can operate in several modes whereby groups of cells can perform similar operations. The size and composition of these groups as well as the mode of operation can be changed dynamically, by switching contexts during runtime. The operation of the RC Array is coordinated by the mRISC Core Controller, which selects and switches contexts at runtime.

3. Development Tools for the MS1 Core

Programs for the MS1 Core consist of two parts: context code and mRISC Core Controller code. Both components are automatically generated from a single Morpho-C program, which is a C/C++ program (labeled "User Code" in Figure 2) with special mStatements. mStatements are high-level statements that the programmer embeds into regular C/C++ code in order to control the operation of the MS1 Core. In a Morpho-C program, the programmer can schedule tasks in order to overlap context loading, data input/output and processing, in many different ways. The Morpho-C code is preprocessed by the mTranslator. The output of mTranslator is a plain, pre-processed C program that is recognizable by the GNU gcc compiler. Both the mTranslator and the gcc compiler are part of the tool set provided with the MS1 Core. Therefore, the programming approach for the MS1 is similar to the paradigm employed by today's conventional DSP processors, with additional instructions for the rDSP fabric. It is important to note that no HDL code is generated at any point in this tool chain. This avoids the lengthy and unpredictable tasks of hardware design such as layout, timing closure and verification and validation.

4. The WCDMA AMR Receiver

The WCDMA AMR channel is defined in [2]. It is a channel that supports data rates from 4.75 to 12.2 kbits/s. However, due to algorithm availability, a GSM EFR is used as the speech codec. The GSM EFR vocoder has the same input and the encoded parameters as the AMR-NB vocoder in mode MR122 [3].

A top-level block diagram of the receiver chain is shown in Figure 3. The received IQ data (at 3.84 Mchips/sec x oversampling rate) is first filtered by a 49-tap RRC FIR filter. The baseband signal is then processed by the RAKE receiver, which performs the tasks of despreading and demodulation, along with the correction for channel and transmitter impairments. The output of the RAKE is then demultiplexed to remove the various physical layer control information such as pilot and power control bits, and passed trough the 2nd Interleaver. After the collection of two radio frames, the 1st

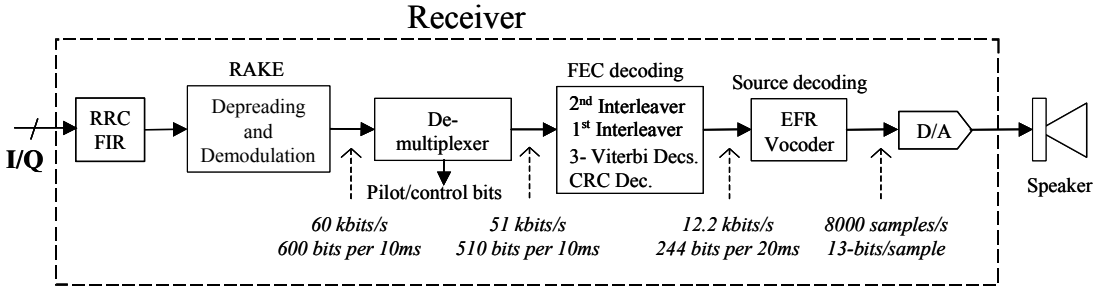

Figure 3: The WCDMA AMR Receiver

interleaver is performed on the received data, after which some control bits (DTX) are removed. The received encoded bits are then processed by the three Viterbi decoders to correct any possible errors caused by the channel. CRC parity decoding is performed on the decoded bits to detect any uncorrected errors on this class of data. The data stream is demultiplexed to its original state at 12.2 kbits/s. These bits are then processed by the GSM EFR decoder, to recover the original PCM speech frame. A Digital-to-Analogue Converter (DAC) processes the audio output signal, and the analog audio signal is played through a speaker. The above mentioned tasks are performed in a regular, time multiplexed schedule within one symbol duration (or Spreading Factor) of 256 "chips" on the RC array.

4.1 RAKE Receiver

The overall RAKE receiver architecture is shown in Figure 4. Each RAKE finger demodulates one signal path. The outputs of all the fingers are combined as a weighted sum in order to maximize the Signal-to-Noise Ratio (SNR).

The internal functions of each RAKE finger are shown in Figure 4. As shown, channel estimation, and clock and carrier synchronization are performed independently for each finger. An external path-searching algorithm provides the necessary timing information to select the optimum sampling instant prior to decimation. With input samples at 2x the chip rate, the Delay Locked Loop (DLL), used for clock synchronization, is capable of correct synchronization, in the presence of worst case expected noise and interference, with an initial acquisition error in excess of ½ chip period. After timing correction by interpolation, the received samples are decimated by a factor of 8, for subsequent chip rate processing by the correlation/despreader units. Soft output data are generated for the Dedication Physical CHannel (DPCH) and the Common PIlot CHannel (CPICH), at the corresponding symbol rates. Early and late correlation values are also generated for the CPICH channel at ±½ chip periods away from the current (punctual) sampling instant. These are used for timing synchronization purposes.

5. Mapping of the Receiver

Starting with a MATLAB simulation model, the receiver and transmitter were validated for functionality and performance. Following that, a fixed point C model was written and was used as the starting point for the mapping. The mapping consists of two major tasks: the first one is to partition the system onto code that runs on the mRISC controller and code that exercises the RC array. The second task is to map the latter code onto the RC array in order to maximize performance and/or reduce code size. The receiver was mapped using a meet-in-the-middle strategy. A first-cut partitioning

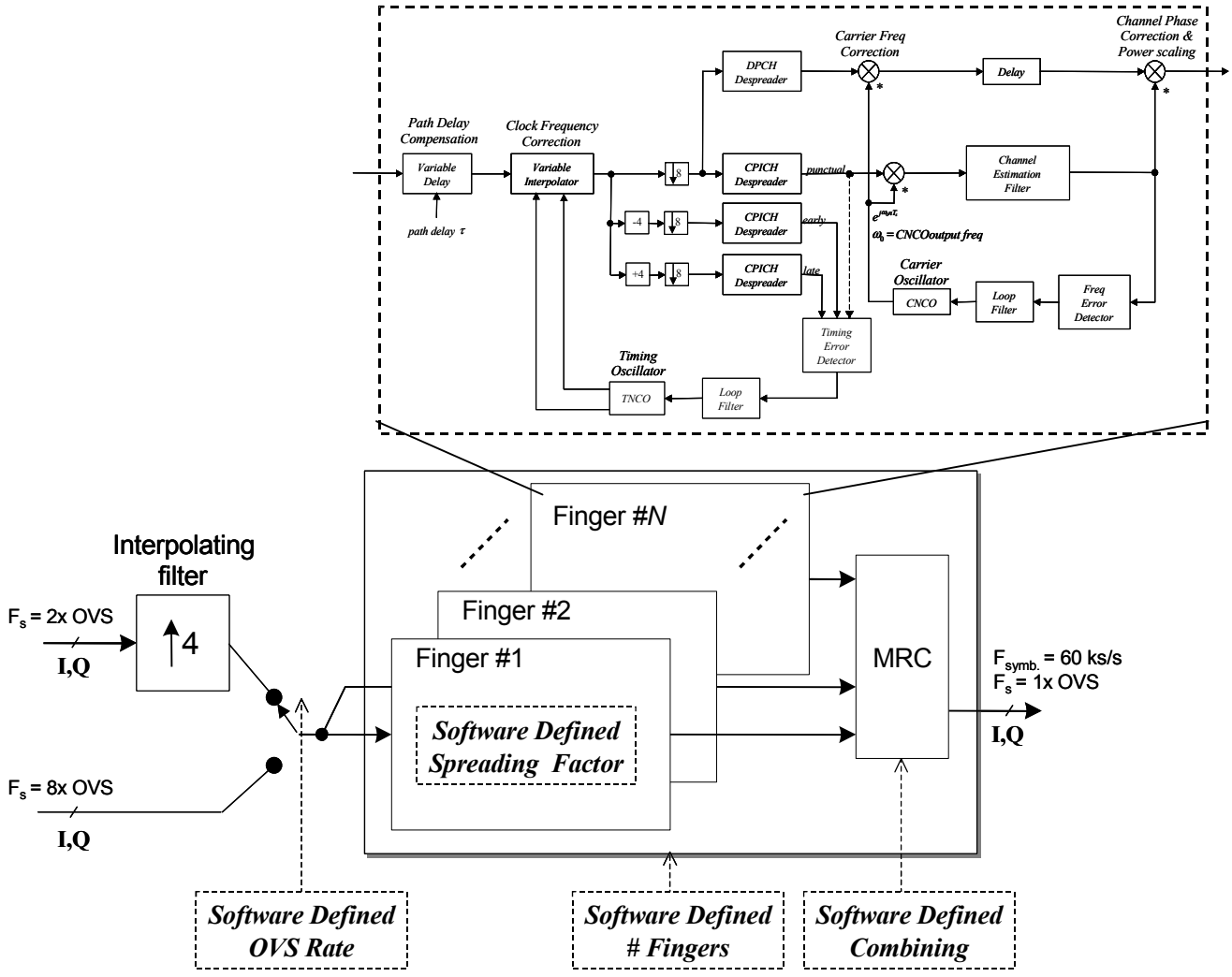

Figure 4: The RAKE receiver

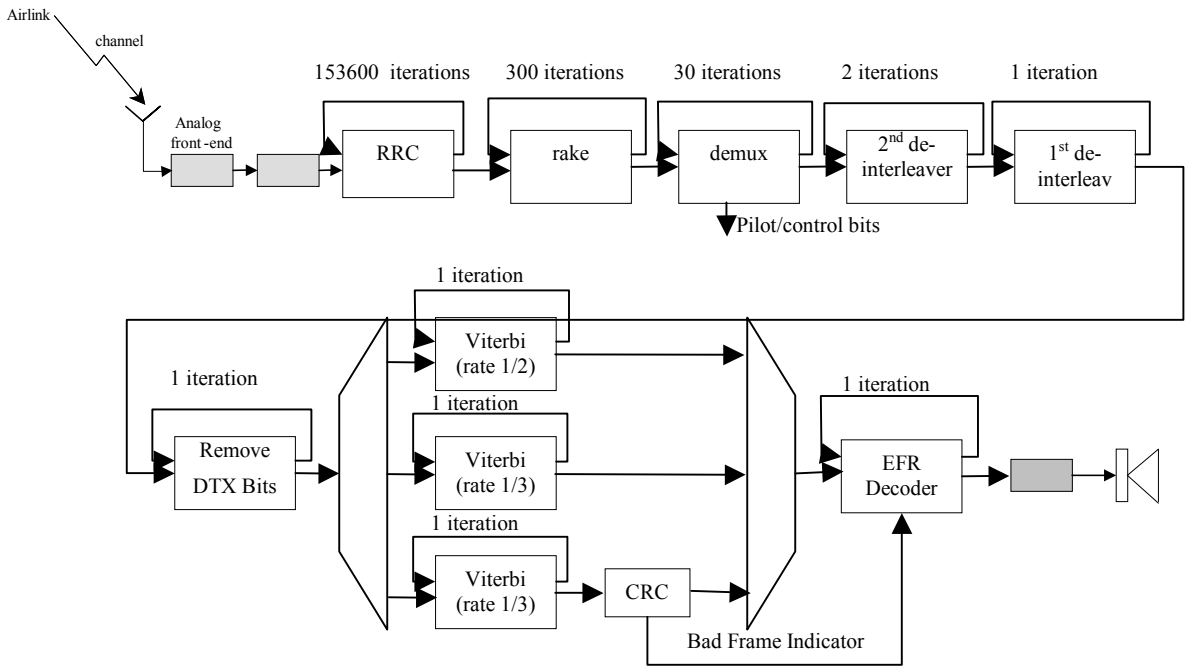

Figure 5: Process Diagram of the AMR Receiver

of tasks was performed manually and the major tasks identified as shown in Figure 5. Next, the tasks that are to run on the RC array were further partitioned onto components that were individually mapped as application kernels and encapsulated as parameterized function calls. Next, the system level C code was modified to incorporate the aforementioned kernels. Contrary to an ASIC solution, the receiver tasks are now executing on a single unit (the MS1 core) in a time-multiplexed mode. This necessitated a scheduling of the kernels in order to ensure that all the real-time requirements for each kernel as well as the overall receiver were met. Additionally, the scheduling has to be applied to the configurations (contexts) for each kernel that must be moved into the core at the right time prior to the execution of the target kernel. The system level scheduling of the overall process diagram of the receiver shown in Figure 5 was adjusted to accommodate the varying granularities of kernels. This necessitated the establishment of buffers between blocks. Recall that the RAKE fingers operate at symbol-rate (66.7μsec), the second interleaver at radio frame rate (10 msec), and all the other operations at speech frame rate (20 msec). In order to maintain real-time operation of the receiver, we chose the coarsest granularity of operation, which in this case is 20 msec as the outer loop iteration time. During that time, the RAKE receiver would output 300 symbols while the second interleaver would output two radio frames. Thus, all the iteration counts shown in Figure 5 were set to produce a speech frame. The code was fine-tuned to maximize the cycle performance. Some kernels or portions of kernels were moved between the mRISC controller and the rDSP fabric depending on the relative performance as well as the communication cost between the two. Conceptually as well as practically, the software-defined nature of the receiver made it quite easy to make code modifications, create buffers, or change scheduling at any time. By contrast, an ASIC implementation would have necessitated a respin of the silicon along with all the associated redesign and fabrication costs.

The following sections describe the receiver algorithms and the mapping strategies employed to optimize their performance on the MS1 core.

5.1 EFR Mapping

Consider the GSM Enhance Full Rate (GSM EFR) vocoder which has the same input and the encoded parameters as the AMR-NB vocoder in mode MR122 [3]. The EFR is the third defined vocoder for GSM systems, and belongs to the Code Excited Linear Prediction (CELP) speech compression family. The EFR with a bit rate of 12.2kbits/sec is an Algebraic CELP (ACELP) vocoder, where an excitation signal is built for each pulse, rather than the selection from a codebook as a codevector, as was the case with classical CELP vocoders. In the EFR vocoder, the codevector is defined as ten pulses of 1 or –1, out of 40 optional subframe locations, where the remaining 30 are set to zero. Thus, the search operation is based on finding the best ten locations for the pulses, and establishing whether each pulse is a 1 or a –1. This task is performed four times per 20msec. Prior to the search and determination of the codevector, two other speech analyses are performed. These are Linear Prediction Coding (LPC) analysis and pitch estimation. These are the fundamental components of the encoder that provide the decoder with the ability to reproduce the synthesized speech at the receiver [4]. There are more than 24 functions for the encoder and more than 12 functions for the decoder part provided by the standard C code [5]. Amongst the encoder functions, the search algorithm (search_10i40) by far consumes most of the encoder processing, which is more than 22% of the total required cycle count for the encoder. This function alone has considerable inherent parallelism, and speed-ups of more than 4x have been achieved in executing this function on a 16-element reconfigurable fabric. Other EFR sub-algorithms such as correlation and filtering also exhibit considerable parallelism, which is dealt with in the following discussions.

5.2 FEC Decoder

This algorithm is known as the Viterbi Algorithm (VA) [6]. With each received n-tuples, the decoder computes a metric or measure of likelihood for all paths that could have been taken during that interval and discards all but the most likely to terminate on each node. An arbitrary decision is made if the path metrics are equal. The metrics can be formed using either hard or soft decisions information with little differences in implementation complexity.

Before the VA is mapped onto the MS1 core, it is necessary to identify the possible strategies for optimal performance. One of the major steps in the Viterbi algorithm is to update the *path metrics* (*pm*), by the appropriate *branch metrics* (*bm*) which are executed in a process known as *Add-Compare-Select* (ACS). Executing a VA on a sequential machine would consume too many cycles and must be sped up for acceptable overall real-time performance. In general, additional parallelism can be introduced into the VA at the bit level, the word level and the algorithm level [7]. At the bit level, transformations allow introduction of bit level pipelining, such that the critical path for each implementation runs only through a few bit levels resulting in the possibility of significant increase in the clock rate [8]. At the word level, using algebraic transformations, the two operations of the ACS recursion may be identified to form an algebraic structure called semi-ring which can represent the VA as linear vector recursion [8]. Using this algebraic transformation, the bottleneck of VA (the non-linear feedback in ACS) is eliminated. At the word level, it is also possible to perform the operation of each of the states independently, in parallel. Finally, at the algorithm level, the asymptotic behavior of the VA can be exploited to derive parallel processing architectures [9]. The bit level parallelism can be incorporated into almost any arithmetic logic unit, regardless of the particular choice of parallel architecture. However, the utilization of word level parallelism is highly dependent on the computational fabric and resources. For a parallel processing engine, with a "modest" number of ACS units at each node (e.g. <10), the feasible word level parallel execution is the concurrent state implementations, ideally with one butterfly operation at each node. It is also possible to assign a single state to each processing node, which might lead to excessive intermediate data movement between the nodes. Algorithm level parallelism offers the highest rewards in extracting the maximum parallelism. The only two drawbacks associated with algorithm level parallelism are, first, the increase in processing latency and, second, the overhead associated with the block overlap. There is usually a tradeoff between the two, where the block overlap overhead can be reduced at the expense of greater throughput latency.

5.3 Interleaver/Deinterleaver

Interleaving is used after the FEC encoding, for randomizing the burst of channel errors caused by multipath fading, such that the errors appear as random as an AWGN channel at the input to the FEC decoder. The random appearance of errors is essential for effective operation of the decoder. The deinterleaving operation is required in the receiver to arrange the data into its original sequence, which is usually the opposite operation to that of interleaving.

The channel interleaving block or bit is usually defined by mathematical or operational algorithms (e.g. the WCDMA first and second interleaving algorithm defined in [10]). A parallel processing fabric, especially with reconfigurable interconnect, may be used for limited permutation changes and bit re-ordering operations. However, for long interleaving depths used in channel interleaving little or no advantage can be gained from such platforms.

5.4 RAKE Receiver

Recall the RAKE receiver functionality described in Figure 4 While current RAKE receiver implementations are done either completely or partially in ASIC logic, the MS1 core is capable of performing the complete RAKE functionality in software running on the MS1. As the chip-rate implementation (i.e. the front-end despreading in each finger) of the RAKE receiver is carried out in software, a number of parameters that were traditionally "hard-wired" in ASICs, can be defined by the operating condition and the required overall performance of the system (i.e. Software Defined). For example, as shown in Figure 4, at the expense of more processing, it is possible to reduce the memory requirement of the input sampled data, by operating at twice the chip rate (i.e. 2x OVS) and using an interpolation filter, to achieve the desired time resolution for fine timing adjustments. It is also possible to set, and change at run-time, the desired number of RAKE fingers and the spreading factor of each finger. Finally, Maximal Ratio Combining (MRC) is used for path combining. Alternatively, any other combining scheme can be integrated in the software, and selected "on-the-fly". The most computationally expensive portion (and fortunately, the most amenable to acceleration) is the despreading. The spreading and despreading of data, performed for spread spectrum modulation, exhibit the most regular form of parallelism, where the same operation is carried out on all the data samples, in parallel. For example, in a WCDMA terminal, the despreading operation of a symbol of an AMR channel requires 128 multiplications of the received data with the appropriate complex code and the accumulation of the multiplication results. This operation can efficiently be spatially mapped to a group of processors where the samples in each block are processed in parallel. The complex chips are moved into the processors, where they are multiplied by the despreading codes, and accumulated in each processing element. After the operation on the final block, a final summation is then performed of all the accumulators of the processors. Therefore, algorithms such as despreading, correlation and FIR filtering that can be implemented usinig a similar structure on the on a reconfigurable processor array, should asymptotically be implemented N times faster on N processors compared to conventional sequential DSP engines. The reconfigurable nature of the RC array plays a central role in the efficient execution of these operations whereby the processors may operate in an SIMD fashion when partial sums are accumulated on individual elements, and switching to an MIMD mode when sums are accumulated across elements. The above mapping example utilizes the word-level parallelism inherently present in the correlation algorithm. Due to the immense flexibility of the reconfigurable fabric, it is also possible to devise alternative spatial mappings that utilize the algorithm-level parallelism, where even higher computational gain should be possible.

The other computationally intensive part of the receiver is the interpolation filter. Almost all the above operations are based on a transversal tap-delay structure that exhibits similar word and algorithm level parallelism, as was discussed for FIR filters in the despreading section, resulting in nearly a N times processing advantage (N = the number of RC processors).

The other components of the RAKE receiver deal mainly with synchronization and recovery loops. These loops occur at different granularities and exhibit little parallelism at the word level. Our first implementation was using mostly sequential code which resulted in adequate but not highly optimized performance. But after closer examination, considerable parallelism can be extracted for each loop both at the word and, more importantly, at the algorithm level. For

example, a WCDMA receiver may require up to eight RAKE fingers to run in parallel. Each of these eight fingers requires independent recovery loops that can be executed in parallel on a reconfigurable processing fabric. Further, at the expense of some data latency, each algorithm can be applied to several symbols in parallel.

The final component of the RAKE receiver is the channel estimation. Radio signals that are subject to multipath propagation suffer attenuation from *Multipath Fadings*. In CDMA based air-interfaces, a "path searcher" unit is used to estimate the delay component, while either the common pilot or dedicated pilot symbols are used for the estimation of the attenuation and phase associated with each significant resolvable propagation path. The path searcher of CDMA based systems uses correlation algorithms based on a transversal filter which is highly parallelizable and very similar to the despreading used in the RAKE receiver.

6. Test bed results

Recalling that the RAKE fingers operate at the symbol-rate (66.7μsec), the second interleaver at radio frame rate (10msec), and all the other operations at speech frame rate (20 msec), the test-bed demonstration showed that a unified time multiplexed approach of all the functions, based on the lowest time granularity, is a feasible processing approach for the implementation of real-time software defined systems. It also demonstrated that the rDSP reconfigurable platform is capable of providing the required processing capacity for the AMR receiver algorithms, including both the chip-rate and symbol-rate processing units. The processing requirement of the chip-rate algorithms (mainly the RAKE receiver) is around 70%, and the Symbol rate (the transport layer) is 30%, of the total processing load required by the these functions. Typically, traditional DSPs are only capable of performing the EFR decoding in real time (which is a very small portion of the overall receiver processing), while the rest of the receiver tasks are performed exclusively by ASIC blocks. Overall, a comfortable amount of headroom was left over to accommodate any additional processing needs, or alternatively to power down to conserve power.

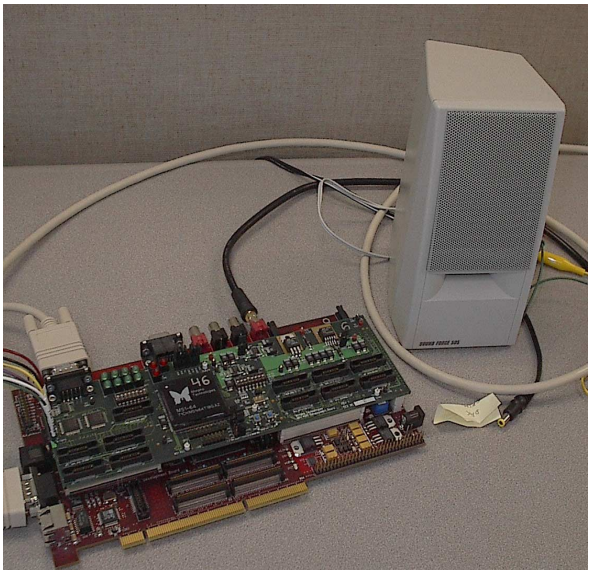

Figure 6: The Hardware Testbed

The AMR receiver was fully implemented using the compilation toolchain described in Section 3 and validated both on a cycle accurate simulator as well as a hardware development system shown in Figure 6. In fact, the test-bed results, and supporting cycle-accurate simulations indicate that all the transmit/receive baseband functions of a handset can be comfortably supported by a single rDSP core with 16 RC processing elements.

While the above discussion was based on a circuit-switched forward-link AMR channel, further benchmarking and cycle-accurate simulations of a 384 kbits/s (Spreading Factor=4) WCDMA channel, supporting packet transmission mode, showed that the same core can support all the transmitter/receiver functions of such channel. The 384 kbits/s channel requires a different RAKE receiver design, to work with relatively shorter transmission duration of packet mode, and supports Turbo encoding/decoding and MPEG codecs. This change is accomplished by simply running a different piece of software than for the previous channel.

7. REFERENCES

[1] W. H. Mangione-Smith *et al*., "Seeking Solutions in Configurable Computing," *IEEE Computer*, Dec. 1997, pp. 38-43

[2] 3GPP TS 34.108, "Common Test Environments for User Equipment (UE) Conformance Testing", Release 4, Technical Specification Group Radio Access Network, 3rd Generation Partnership Project.

[3] 3GPP-AMR-NB With ETSI-EFR Implementation on the StarCore SC140 Core, (AN2280/D), Rev 0, 5/2002, Motorola application note.

[4] 3GPP TS 46.051, "Enhanced Full Rate (EFR) speech processing functions: General description" Release 4, Technical Specification Group Services and System Aspects, 3rd Generation Partnership Project.I

[5] GSM 06.53: "Digital cellular telecommunications system (Phase 2+); ANSI C code for the GSM Enhanced Full Rate (EFR) speech codec".

[6] A. J. Viterbi, "Error bounds for convolutional coding and an asymptotically optimum decoding algorithm," *IEEE Trans. Information Theory*, vol. IT-13, pp. 260-269, April 1967.

[7] G. Fettweis, H. Meyr, "A Modular Variable Speed Viterbi Decoding Implementation for High Data Rates," North-Holland Signal Processing IV, Proc. EUSIPCO '88, pp. 339-342, 1988

[8] G. Fettweis, H. Meyr, "High-Speed Parallel Viterbi Decoding: Algorithm and VLSI Architecture," IEEE Communication, pp. 46-55, May. 1991

[9] G. Fettweis, H. Meyr, "Feedforward Architectures for parallel Viterbi Decoding," Kluwer J. on VLSI signal processing, No. 3 pp. 105-119, 1991.

[10] 3GPP TS 25.212, "Multiplexing and channel coding (FDD)", Release 4, Technical Specification Group Radio Access Network, 3GPP.