Estimating the Computational Requirements of a Software GSM Base Station

Thierry Turletti and David Tennenhouse
{*turletti,dlt*}@ *lcs.mit.edu*
Telemedia Networks and Systems Group
Laboratory for Computer Science, MIT

Abstract

In this paper, we describe the software implementation of a GSM base station and consider the performance of each of its Radio Interface components. The performance of each software module is evaluated using both a % CPU metric and a processor-independent metric based on SPEC benchmarks. These results can be used to dimension systems, e.g., to estimate the number of GSM channels that can be supported by a given processor configuration, and to predict the impact of future processor enhancements on base station capacity.

1 Introduction

Historically, the use of software within radio systems has been restricted to low bandwidth *out-of-band* tasks, such as signaling, operations and management. Implementation of the *in-band* Radio Interface functionality has leveraged analog front ends, that partition the band into individual FDM channels, and digital components that perform the time division and speech processing functions. The design of the digital portion is typically based on some compromise between a full ASIC implementation and programmable DSPs. In particular,past efforts to develop *software radios* have leveraged DSP-based software to emulate functions traditionally implemented in dedicated digital hardware [5, 7].

In contrast, our *virtual radio* approach involves the reexamination of the overall system from a software designer's perspective, with specific emphasis on the *virtualization* of RF sources. Advances in processor[1] and A/D conversion technology [3], combined with recent improvements in memory and I/O bandwidth, have enabled a complete software solution. In particular, wideband A/D technology can be used to sample an entire RF band and the resultant sample stream can be directly deposited into memory for software-based analysis. This eliminates the need for dedicated per-channel frequency partitioning hardware. Furthermore, the temporal decoupling of the sample stream, using large memory buffers, allows conventional operating systems and networks to bridge the gap between the A/D converter and the software processing modules. Accordingly, both the frequency partitioning and the digital processing (previously performed by ASICs and DSPs), can be implemented in portable software modules that execute on general-purpose workstations. This approach is being pioneered by the SpectrumWare[2] project whose aim is to extend the reach of application software as close as possible to the A/D boundary. [8, 9].

In this paper, we focus on the design and performance of a library of software modules that can be used to implement the *in-band* portions of a GSM base station. GSM, the Global System for Mobile communications, is a digital cellular telephony system that has gained world-wide acceptance.[3]. Besides its widespread adoption, we have chosen GSM as an experimental vehicle because: its complexity is high enough to generate a variety of interesting problems; it is a real system whose implementations are worthy opponents, i.e., they use state of the art analog and DSP technology; and its design is well documented.

The structure of the paper is as follows. In section 2 we briefly describe the hardware and the software environment in which we are developing the software base station. Sections 3 and 4 describe the implementation of the sending side and the receiving side of the base station, respectively, and evaluate the computational requirements of each of the individual modules. In section 5, we discuss the overall computational requirements of a complete base station and a potential system configuration for its realization.

Except where specifically noted in the text, little time has been spent optimizing the code of the individual modules, so it is likely that the reported performances can be improved. On this basis, we suggest that, in the not too distant future, it will be possible to implement a *mini-cell* that supports 7-15 speech channels, i.e., 1-2 FDM channels, using a commercially available multi-processor server or a small cluster of workstations. We specifically identify the modulation, equalization and demodulation modules as those most in need of further development and optimization.

Two novel aspects of this work are the portability of the software modules and the platform-independent evaluation of their computational requirements. For each module, we first determine the fraction of CPU it requires when executing on a range of workstation platforms. We then normalize these individual results with respect to the known performance of the workstations, using the industry standard SPEC benchmarks. This gives us an estimate of the module's computational requirements in SPECmarks. Although this is far from a precise characterization, it has the benefit of being platform-independent. Furthermore, it allows us to predict module performance on future platforms, based on anticipated improvements in the SPECmark ratings of next generation workstations and servers.

In summary, we believe that the software realization of a GSM base station is becoming a viable alternative to present designs. A software only approach has the potential of offering many benefits: a software base station can support several modulation schemes and otherwise incompatible standards (e.g. AMPS, TDMA IS-54, CDPD, GSM). Furthermore, the elimination of dedicated hardware reduces the complexity of the base station and introduces tremendous flexibility and platform inde-

[1] The available cpu power on general purpose processors doubles every about 18 months.

[2] See URL: "http://www.tns.lcs.mit.edu/SpectrumWare/".

[3] By the middle of 1995, 75 networks were in service in 45 countries, serving around 8 million subscribers.

pendence into the overall system.

2 System Overview

In this section we give a brief description of the minimal hardware required to support a software base station and we describe the software environment in which we are developing the GSM modules.

2.1 Simple wideband RF transceivers

Wideband A/D conversion allows the construction of simple RF transceivers, consisting of only: an antenna, a pre-amplifier, a bandpass filter, and a wideband A/D converter in the receive path (see Figure 1); and a D/A converter and an amplifier in the transmit path.

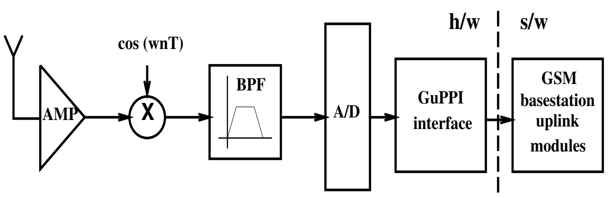

Figure 1: Minimal hardware for a GSM receiver base station

The sampled signal data is directly transferred into the memory of the workstation/server via a DMA interface, such as the General Purpose PCI Interface (GuPPI) developed at MIT. This PCI expansion board provides an efficient means for the continuous streaming of sample data to/from the host memory. Recent performance measurements show data rates of around 80 Mbytes/sec from host memory to the D/A interface and 90 Mbytes/sec from the A/D interface into host memory. Assuming that samples are stored in memory as 16 bit units, this interface will support 40 M-samples/second in either direction. Although not quite sufficient to capture the entire 25 MHz GSM band, planned enhancements to the PCI bus will quadruple the present I/O capacity.

2.2 The Software Environment

Our prototype software is based on the VuSystem, a programming system for the dynamic manipulation of temporally sensitive data [6]. The VuSystem implements a two level strategy, in which programs are partitioned along an *in-band* axis, that supports the flow of temporally-sensitive information, and an *out-of-band* axis that supports the event driven program components, including the user interface and the configuration and control of the in-band processing pipeline. In-band processing and out-of-band processing are best handled in separate partitions instead of together, because then choices of language and architecture can be made for each partition separately (e.g. C++ for in-band and Tcl for out-of-band processing). The in-band processing partition is arranged into processing *modules* which logically pass dynamically-typed data *payloads* through input and output ports. The payload abstraction is an efficient way to hide the implementation details such as shared-memory data regions, so that the designer of in-band processing modules needs only to know a few simple rules about payload handling.

In the following sections, we describe the design of the in-band [4] portion of a GSM base station. We have classified the modules into two classes: the downlink modules that correspond to the sending side of the base station, and the uplink modules corresponding to functions performed in the reverse path. For each module, we discuss briefly the functionality provided, the algorithm we have chosen, and the performance obtained.

[4]Currently, we have not implemented any signaling functions.

2.3 Estimation of Computational Requirements

To measure performance, we have isolated each module and run it separately in a loop with a 20 ms input chunk of data. Execution times on a number of different platforms are measured using the shell *time* function. We then computed the corresponding percentage of each platforms' cpu that the module consumes when performing the processing associated with a single GSM speech channel.

The resultant"% cpu / GSM channel" is a good metric to report performance of each module on a given platform. However, these results will rapidly become outdated with the ongoing introduction of new generations of processors. Quantification of computational requirements in a processor-independent way is critical for a vision of *plug-and-play* telecommunications. Accordingly, we have adopted the SPEC[5] benchmark as a vehicle for the normalization of our results. In particular, we normalize each of the % cpu results with respect to the SPEC92 and SPEC95 ratings of each platform, as presented in Tables 13 and 14, respectively. This gives us an estimate of the module's computational requirements in SPECmarks. Although the mapping to a module's SPECmark requirement is not entirely platform-independent we are able to obtain a SPECmark range that can provide a basis for performance predictions. This generic metric is not precise but we believe that it is nonetheless an improvement on platform-dependent measures of computational requirements.

3 From Speech to Radio (Downlink)

The sequence of downlink modules is shown in Figure 2. Basically, the GSM speech coder compresses speech into digital blocks. Channel coding adds redundancy to the blocks which are then interleaved and spread into pieces called bursts. Finally, after the ciphering operation, the bursts are used to modulate the phase of a carrier.

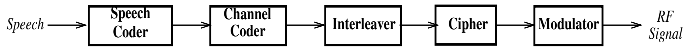

Figure 2: A block diagram of a GSM sender base station (downlink)

3.1 GSM speech coding

The GSM full rate speech coding algorithm is called RPE-LTP, which stands for Regular Pulse Excitation with Long Term Prediction [2]. Basically, the analog voice is first digitized in an A/D converter to 8000 samples per second, uniformly coded at 13 bits each. Then, the encoder divides the speech into a short-term predictable part, a long-term predictable part and the remaining residual pulse. Finally, it encodes the residual pulse and parameters describing the two predictors. The speech coding algorithm produces a speech block of 260 bits every 20 ms. These 260 bits are classified into three classes (Ia, Ib and II) according to their importance.

We are using a software implementation of this vocoder done by Jutta Degener and Carsten Bormann from the Technical University of Berlin[6]. The performance[7] of this vocoder is shown in Table 1 on a number of platforms.

[5]SPEC stands for System Performance Evaluation Cooperative, see URL "www.specbench.org/spec/".

[6]The implementation is available in the public domain, see URL: "www.cs.tu-berlin.de/˜jutta/toast.html".

[7]All C and C++ modules are compiled with gcc v2.7.2 using the -O2 optimization flag.

Speech Coder (GSM 06.10 RPE-LTP)	Perf. (metric/user-channel)				
	% cpu	SPECint		SPECfp	
		92	95	92	95
SS 10/51	20	13	–	17	–
SUN Ultra M170	8.1	20	0.45	28	0.73
DEC 3000/800	10	14	–	19	–
Intel Pentium 166	10	20	0.48	–	0.34

Table 1: GSM speech coding performance

3.2 Channel coding

Class Ia bits are first protected by 3 CRC bits for error detection. The Class Ib bits are then added to this result. A convolutional code with rate $r = \frac{1}{2}$ and constraint length $K = 5$ is applied to this complete class I sequence. The resulting 378 bits are used in conjunction with the 78 unprotected Class II bits, to form a complete coded speech frame of 456 bits (see Figure 3).

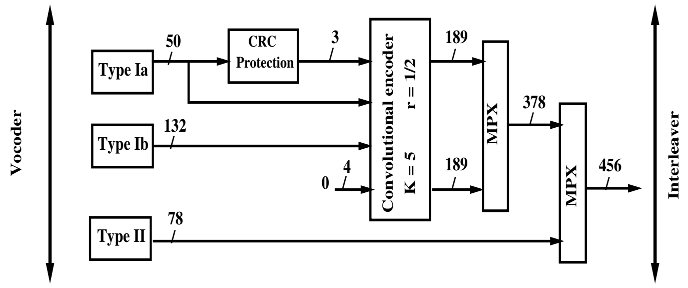

Figure 3: TCH/FS Transmission Mode

Error detecting codes

As indicated, the most sensitive portion of the speech frame (i.e. the 50-bits category Ia) are protected by three CRC bits used for error detection. These bits are computed using the polynomial generator[8]: $G(X) = X^3 + X + 1$.

Convolutional coding

The convolutional step involves the padding out of the class I sequence with 4 extra bits and a convolution using two polynomials. The result of this step is two 189 bit sequences, which are multiplexed with the (unprotected) type II bits to yield a 456 bit coding block, see Figure 3.

We have implemented the channel coding algorithm and its performance is summarized in Table 2.

Channel Coding	Perf. (metric/user-channel)				
	% cpu	SPECint		SPECfp	
		92	95	92	95
SS 10/51	0.43	0.28	–	0.36	–
SUN Ultra M170	0.20	0.50	0.011	0.70	0.018
DEC 3000/800	0.45	0.73	–	0.85	–
Intel Pentium 166	0.38	0.76	0.018	–	0.013

Table 2: Channel coding performance

[8] Note that the "+" operation stands for the XOR logical operation.

3.3 Interleaving

The aim of interleaving is to decorrelate the relative positions of the coded bits within the code words and in the modulated radio bursts. GSM coding blocks are interleaved on 8 bursts: the 456 bits of one block are split into 8 groups of 57 bits. Each group of 57 bits is carried in a different burst.

The major drawback of interleaving is the corresponding delay: transmission time from the first burst to the last one in a block is equal to 65 burst periods (i.e. 37.5 ms). However, from a software perspective, the performance of our implementation is shown in Table 3.

Interleaver	Perf. (metric/user-channel)				
	% cpu	SPECint		SPECfp	
		92	95	92	95
SS 10/51	0.19	0.12	–	0.16	–
SUN Ultra M170	0.04	0.11	0.002	0.15	0.004
DEC 3000/800	0.12	0.17	–	0.23	–
Intel Pentium 166	0.15	0.30	0.007	–	0.005

Table 3: Interleaving performance

3.4 Ciphering

Ciphering is achieved by performing an XOR (exclusive OR) operation between a pseudo-random bit sequence and the 114 bits of each burst. The pseudo-random sequence is derived[9] from the burst number and a session key established through signaling when a call is established. The performance of the algorithm is shown in Table 4.

De/Cipher	Perf. (metric/user-channel)				
	% cpu	SPECint		SPECfp	
		92	95	92	95
SS 10/51	0.24	0.16	–	0.20	–
SUN Ultra M170	0.08	0.19	0.004	0.26	0.007
DEC 3000/800	0.16	0.22	–	0.30	–
Intel Pentium 166	0.09	0.17	0.004	–	0.003

Table 4: Ciphering and Deciphering performance

3.5 Modulation

GSM uses Gaussian Modulation Shift Keying (GMSK) with modulation index $h = 0.5$, BT (filter bandwidth times bit period) equal to 0.3 and a modulation rate of 271 kbaud. The Gaussian filtering function represents the most cpu intensive part of the algorithm. The current version of the modulator uses an IIR implementation and only supports an integral number of samples per symbol (FsT). Note that the complexity of the algorithm is proportional to the FsT parameter. Table 5 reports performance obtained with FsT=2 (which corresponds for GSM to a sampling rate 2.7 times higher than the maximal signal frequency).

4 From Radio to Speech (Uplink)

The sequence of uplink modules are shown in Figure 4. These modules perform the following operations: the demodulator reconstitutes the GSM bursts. Then, after the deciphering and the

[9] Note that the algorithm used to generate the pseudo-random sequence is not fully included in the public GSM specification.

GMSK Modulator	Perf. (metric/user-channel)				
	% cpu	SPECint		SPECfp	
		92	95	92	95
SS 10/51	69.0	45.0	–	57.3	–
SUN Ultra M170	30.4	76.6	1.69	107	2.75
DEC 3000/800	25.6	35.4	–	48.0	–
Intel Pentium 166	21.9	43.3	1.04	–	0.74

Table 5: GMSK Modulator performance

de-interleaving operations, the channel decoder tries to recover the transmitted blocks of data. Finally, a GSM speech decoder converts them into audio samples.

Figure 4: A block diagram of a GSM receiver base station (uplink)

4.1 Channel partitioning

The number of frequency channels used by a GSM base station depends mainly on the density of the network and the type of cells used (micro cells, urban, rural or road cells). It can be 1 in low density area, 2 to 4 in high density area. Actually, adjacent channels are not used within the same cell to avoid spectral recovering. Typically, a GSM base station uses every second channel (i.e. 200 kHz spacing) to avoid high level interferences. In the GSM terminology, a logical channel corresponds to a speech channel (defined by its frequency and its time slot number) whereas a physical channel stands for a (FDM) channel (which contains 8 logical channels).

Multichannel receivers for digitized data can be synthesized using a polyphase transform algorithm [3]. Basically, this algorithm performs the following tasks: (1) frequency translation (each center frequency to baseband); (2)bandwidth reduction of the translated spectrum to match the signal bandwidth; (3) resampling of the output to match the reduced channel bandwidth.

Table 6 reports the percentage of cpu per platform required to partition N GSM frequency channels ($N \in [1, 8]$) at a sampling rate equals to 2.5 times the maximal frequency and with a 200kHz spacing scheme. The filter used in the implementation has 255 coefficients. The corresponding SPEC performances are shown in section 5.1, Table 12. [10]

The high CPU requirement[11] is due to the huge input data rate and the need to compute imaginary and real components of output samples for each frequency channel (floating point operation).

4.2 Demodulation and Equalization

The GSM specifications do not impose one particular demodulation algorithm. However, they impose minimal performance criteria and the algorithm used is expected to cope with two multipaths of equal power received at intervals of up to 16 μs (i.e. more than four symbols). With such a level of intersymbol interference, simple demodulation techniques are ineffective and an equalizer is required. We are currently implementing an

[10] Note that the SPECint numbers are not very significant since the partitioning algorithm mainly executes floating-point multiplications.

[11] The software implementation of this algorithm has been optimized in C.

Polyphase Transform	Perf. (%cpu)			
# frequency channels	1	2	4	8
SS 10/51	60	180	610	2200
SUN Ultra M170	20	60	180	620
DEC 3000/800	25	80	220	670
Intel Pentium 166	40	110	340	1200

Table 6: Polyphase Transform performance (200 kHz spacing between channels)

equalizer based on the Viterbi algorithm [1] and this module is expected to be processor-intensive[12].

4.3 Deciphering

Deciphering performs the same operation as the ciphering module described in section 3.4.

4.4 De-interleaving

De-interleaving performs the inverse operation to the interleaving described in section 3.3. Every 40 ms, eight 114-bits bursts are merged into a 456-bit buffer. The performance of this module is shown in Table 7.

De-interleaver	Perf. (metric/user-channel)				
	% cpu	SPECint		SPECfp	
		92	95	92	95
SS 10/51	0.11	0.072	–	0.091	–
SUN Ultra M170	0.045	0.11	0.003	0.16	0.004
DEC 3000/800	0.17	0.24	–	0.32	–
Intel Pentium 166	0.16	0.32	0.008	–	0.005

Table 7: De-interleaving performance

4.5 Channel decoding

Channel decoding involves the retrieval of the original compressed speech data from the (possibly corrupted) received flow.

Figure 5 shows the different steps of the channel decoding algorithm. The de-interleaving algorithm generates a 456-bit data buffer from eight GSM bursts. Some of these bits (the 378 bits corresponding to the Type I data bits) are fed into the convolutional decoder, which tries to reconstitute the 189-bits corresponding to the original sequence. After a bit-reordering step, the 50-bits that have the highest priority (Type Ia) are checked using an error control algorithm . If there is no error, a final bit reordering is performed. These steps are further described in the following paragraphs.

Convolutional decoding

The convolutional decoding can be performed using the Viterbi algorithm. The encoder memory is limited to K bits; a Viterbi decoder in steady-state operation tacks only 2^{K-1} paths. Its complexity increases exponentially with the constraint length K, which is equal to 5 for the GSM convolutional code, see section 3.2.

[12] It has previously been shown that a GSM base station equalizer can be implemented using a full TMS320/C40 DSP, i.e. consuming 50 MIPS.

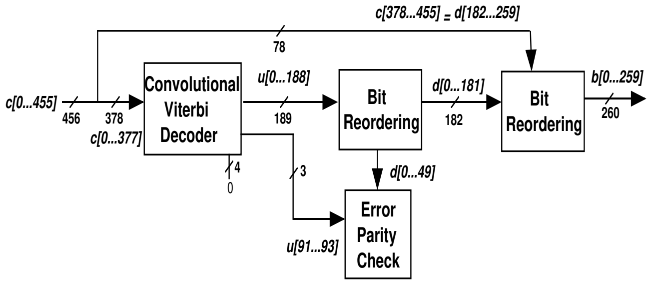

Figure 5: GSM channel decoding

Our implementation is a modified version of the Viterbi decoder[13] developed by Phil Karn for the NASA standard code (K=7, r=1/2) [4].

Reordering algorithm

The 189 bits generated by the Viterbi decoder ($u(0), u(1), ..u(188)$) consists of 182 information bits of class 1 ($d(0), d(1), ..d(180)$), 3 CRC-bits and 4 tail bits.

Error detecting codes

The polynomial operation described in section 3.2 is applied to the CRC-protected data. If the remainder differs with the received CRC bits, an error is detected and the audio frame is tossed and eventually discarded.

Reordering algorithm

The last step of the channel decoding regroups the 260 bits of the speech block from the position that corresponds to the order of decreasing importance ($d(0), d(1), .., d(259)$) to the position that matches their classification ($b(0), b(1), .., b(259)$) as specified in [2]. This operation (which requires no effort while implemented in hardware) can be efficiently implemented in software using look-up tables.

The performance of the channel decoding algorithm is summarized in Table 8.

Channel Decoder	Perf. (metric/user-channel)				
	% cpu	SPECint		SPECfp	
		92	95	92	95
SS 10/51	4.2	2.7	–	3.5	–
SUN Ultra M170	1.7	4.2	0.094	5.9	0.15
DEC 3000/800	2.2	3.1	–	4.2	–
Intel Pentium 166	2.1	4.2	0.099	–	0.070

Table 8: Channel decoding performance

4.6 GSM speech decoding

The decoder reconstructs the speech by passing the residual pulse first through the long-term prediction filter, and then through the short-term predictor. As for the GSM speech coder, the decoder has been implemented by Jutta Degener and Carsten Bormann, see section 3.1. The performance of this module is shown in Table 9.

[13] See URL "http://www.qualcomm.com/people/pkarn/ham.html".

Speech Decoder (GSM 06.10 RPE-LTP)	Perf. (metric/user-channel)				
	% cpu	SPECint		SPECfp	
		92	95	92	95
SS 10/51	10	6.7	–	8.5	–
SUN Ultra M170	4.4	11	0.24	15	0.40
DEC 3000/800	5.2	7.2	–	9.8	–
Intel Pentium 166	3.9	7.7	0.19	–	0.13

Table 9: GSM 06.10 decoding performance

5 Putting it all Together

In this section we discuss: the overall computational requirements of a hypothetical base station configuration; and our efforts to integrate the modules into an experimental platform.

5.1 Estimating a Base Station's Computational Requirements

Table 10 summarizes the performance of the modules required to encode and decode one GSM logical(TDM) channel, exclusive of the modulator, demodulator/equalizer and partitioner modules. We give the minimum and maximum SPEC numbers we have obtained with the four different platforms.

In Table 11 we project the computational requirements for the encoding and decoding of N GSM physical (FDM) channels, ($N \in [1, 8]$), where each physical channel supports 8 logical channels. Here again, the table entries are exclusive of the partitioner, modulator and demodulator/equalizer modules. Furthermore, it is assumed that, for each groups of up to four physical channels, one of the derived logical channels is dedicated to signaling and the remaining logical channels require speech processing. This table suggests that a single Sun UltraSparc M170 workstation can already support the encoding/decoding requirements of 1 physical GSM channel.

Table 12 reports the SPEC-normalized performance of the partitioning algorithm discussed in section 4.1. The results suggest that our current M170 platform would be sufficiently powerful to perform the partitioning of two physical (FDM) channels.

Taken together, the above tables suggest that a *mini-cell* base station supporting two physical channels, i.e., up to 15 logical channels could be assembled using: one workstation to perform the partitioning of incoming samples; one workstation to multiplex the outgoing sample streams; two workstations to perform the encoding/decoding; and some number of additional processors to realize the modulation/equalization/demodulation stages. Although our initial implementations of these modules are processor-intensive, we believe it will be possible to realize a complete two channel system using a cluster of 10-15 workstations, or possibly, a single multiprocessor server. Although this seems rather expensive today, improvements in processor technology suggest that this level of functionality will be available in a single workstation or PC within 4-5 years.

5.2 Module Integration

With the exception of the equalizer, which is currently being implemented, we have developed prototype C/C++ implementations of the modules described in this paper. These modules have been incorporated into the VuSystem programming environment, which supports their integration into experimental applications. Our next steps will be to tune the performance of the individual modules and to experiment with the system as a whole. We are moving towards *real-world* experiments, initially using an RF Channel Simulator, such as the HP 11759c. Eventually, we hope to demonstrate inter-operation of the uplink modules with a production handset. Two key questions to be addressed involve: the

GSM Modules (for one GSM logical channel)	Performance (metric/user-channel)							
	SPECint92		SPECfp92		SPECint95		SPECfp95	
	min	max	min	max	min	max	min	max
Speech coder	13	20	17	28	0.45	0.48	0.34	0.73
Channel coder	0.28	0.76	0.36	0.85	0.011	0.018	0.013	0.018
Interleaver	0.11	0.30	0.15	0.23	0.002	0.007	0.004	0.005
Cipher	0.16	0.22	0.20	0.30	0.004	0.004	0.003	0.007
Decipher	0.16	0.22	0.20	0.30	0.004	0.004	0.003	0.007
Deinterleaver	0.07	0.32	0.091	0.32	0.003	0.008	0.004	0.005
Channel decoder	2.7	4.2	3.5	5.9	0.094	0.099	0.070	0.15
Speech decoder	6.7	11	8.5	15	0.19	0.24	0.13	0.40
Total	23	37	30	51	0.76	0.86	0.57	1.3

Table 10: GSM modules performance

Base Station # frequency channels	# log. ch data/total	SPECint92		SPECfp92		SPECint95		SPECfp95	
		min	max	min	max	min	max	min	max
1	7/8	180	300	240	410	6.1	6.9	4.6	10
2	15/16	370	590	480	820	12	14	9.1	21
4	31/32	740	1180	960	1630	24	28	18	42
8	62/64	1470	2370	1920	3260	49	55	36	83

Table 11: GSM Base Station performance

multi-tasking and communication overhead associated with the concurrent operation of the entire suite of modules; and the latency characteristics of the software-based system. The latter may be less of an issue here than in other domains, given the built-in latency of the GSM speech coder.

6 Related Work

The Speakeasy Multiband Multimode Radio (MBMMR) is the most aggressive implementation of a software radio [5]. That project's objective was to emulate more than 15 existing military radios, operating in frequency bands between 2 and 200 MHz by using DSPs. However, the effort put into the DSP processor design coupled with the low-level programming environment limit the portability of the system to other processing platforms.

A computer simulation of the Radio Channel Aspects of the GSM System , based on the MATLAB[14] toolkit has been implemented by David T. Margrave[15]. The purpose of this simulation package is strictly educational, demonstrating the necessity of adaptive channel equalization and channel coding operations of the GSM system. However, such simulations do not have real-time behavior.

7 Conclusions and Future Work

In this paper we have presented the design and implementation of the in-band components of a software-only realization of a GSM base station. We have evaluated their computational requirements of each of the components and identified the modulation, equalization and demodulation stages, as the key targets for optimization and tuning.

We have introduced the SPEC benchmark metric as a platform-independent means to estimate the requirements of the individual modules. Although this method is not perfect, we believe it provides a very useful way to benchmark the algorithms and to forecast module performance on future platforms. We are not aware of previous use of the SPEC metrics as a means of benchmarking software. Our approach allows us to project the impact of software-based processing on wireless communications. In 4-5 years it may be possible to implement a complete base station on a single workstation or small server. This suggests that organizations embarking on new design efforts should be cognizant of the software-only approach. As processor and A/D technology continue to improve, it will be possible to implement increasingly sophisticated and flexible processing on ever wider bands of sampled RF spectrum.

The software approach provides tremendous flexibility for mobile communications systems. For example, changing modulation schemes or frequency assignments only involves the loading and/or configuration of software modules. Similarly, the base station software can be upgraded to support new modulation schemes without having to upgrade hardware, especially the RF components. At the same time, the scheme preserves the vendor's software investment, as modules can be ported to higher performance platforms, e.g., to support increased cell capacity. Finally, we note that the software-only strategy can also be applied to the design of the mobiles – units moving between service providers, or continents, can be dynamically configured to interoperate with local facilities.

We are currently working on improving the performance of the key modules and investigating the use of so-called multimedia processor extensions, including Intel's MMX and Sun's VIS instructions. We are also working towards an integrated demonstration of a complete mini-cell base station configuration.

Acknowledgments

This work was supported by the ARPA under contract No. F30602-92-C-0019 (monitored by AFSC, Rome Laboratory) and contract No. DABT-6395-C-0060 (monitored by US Army, Fort Huachuca). As part of his master, Hans Bentzen optimized the GMSK modulator and worked on the equalization module. He provided the latest performance results of the GMSK modulator.

[14] MATLAB is a registered trademark of the MathWorks, Inc.

[15] See URL: "http://www.utw.com/~dmargrav/gsm/ece798/paper.html".

Base Station	SPECint92		SPECfp92		SPECint95		SPECfp95	
# frequency channels	min	max	min	max	min	max	min	max
1	35	80	47	70	1.1	1.9	1.4	1.8
2	110	220	150	210	3.3	5.2	3.7	5.4
4	300	670	410	630	10	16	11	16
8	930	2400	1300	2300	34	57	40	56

Table 12: Partitioning performance

Platform	Clk (MHz) ext/in	Cache ext+I/D	SPECint 92	SPECfp 92	Info date	Source obtained
SS 10/51	40/50	1M+20/16	65.2	83.0	Apr93	Sunflash
SUN Ultra M170	83/167	512+16/16	252	351	Nov95	SunIntro
DEC 3000/800	40/200	2M+8/8	138.4	187.6	May94	c.sun.hw
Intel XXpress Pentium	66/166	1M+8/8	197.5	–	Jan96	www.intel

Table 13: SPEC92 Benchmarks

Platform	Clk (MHz) ext/in	Cache ext+I/D	SPECint 95	SPECfp 95	Info date	Source obtained
SS 10/51	40/50	1M+20/16	–	–	–	–
SUN Ultra M170	83/167	512+16/16	5.56	9.06	Mar96	c.bmarks
DEC 3000/800	40/200	2M+8/8	–	–	–	–
Intel XXpress Pentium	66/166	1M+8/8	4.76	3.37	Jan96	www.intel

Table 14: SPEC95 Benchmarks

A SPECmarks for several machines

Tables 13 and 14 report respectively SPEC92 and SPEC95 benchmarks for the four platforms[16] used in our experiments.

References

[1] Forney, G.D. JR. "The Viterbi Algorithm" *Proc. of the IEEE*, Vol. 61, No 3, Mar. 1973, pp. 268-278.

[2] "GSM 06.10 - European digital cellular telecommunications system (Phase 2); Full rate speech transcoding", *ETS 300 580-2, European Telecommunication Standard*, Sept. 1994.

[3] Harris, F. and Steinbrecher, D. "Wireless Interface Design for Digital Processors", *In ICUPC'93 Tutorial 5*, Oct. 1993.

[4] Karn, P. "Convolutional decoders for amateur packet radio", *Proc. of ARRL'1995 Digital Communications Conference*, 1995.

[5] Lackey, L. and Upmal, D. "Speakeasy: The Military Software Radio", *IEEE Communications Magazine*, pp. 56-61, Vol. 33, No. 5, May 1995.

[6] Lindblad, C. J., Wetherall, D. and Tennenhouse, D. L. "The VuSystem: A Programming System for Visual Processing of Digital Video", *Proc. of ACM Multimedia 94*, San Francisco, CA, Oct. 1994.

[7] Mitola, J. "Software Radios Survey, Critical Evaluation and Future Directions", *Proc. IEEE National Telesystems Conference*, Washington, DC, May 19-20, 1992.

[8] Tennenhouse, D.L., and Bose, V.G. "SpectrumWare - A Software-Oriented Approach to Wireless Signal Processing", *Proc. ACM Mobile Computing and Networking 95*, Berkeley, CA, Nov. 1995.

[9] Tennenhouse, D.L., Turletti, T. and Bose, V.G. " The SpectrumWare Tesbed for ATM-based Software Radios", *Proc. of ICUCP'96*, Boston, Sept. 29 - Oct. 2 1996.

[10] Turletti, T. "A brief Overview of the GSM Radio Interface", *Technical Memorandum, TM 547*, MIT, Mar. 1996.

[16] Some SPECmarks are missing in the list of benchmarks available in the public domain, see URL "www.specbench.org/osg/cpu95/results/".