GSM/GPRS PHYSICAL LAYER ON SANDBLASTER DSP

Raghunath Kalavai, Murugappan Senthilvelan, Sitij Agrawal, Sanjay Jinturkar, John Glossner
Sandbridge Technologies, 1 North Lexington Avenue, White Plains, NY 10601
rraghunath@sandbridgetech.com

ABSTRACT

Handsets are converging to multimedia multi-protocol systems with multimedia and communications systems. Sandbridge Technologies has developed the multithreaded Sandblaster DSP core for such convergent devices. This paper presents different approaches to the implementation of GSM/GPRS physical layer on the Sandblaster DSP. Testing of real time performance with a commercial RF front-end is also presented.

1. INTRODUCTION

GSM (Global System for Mobile Communications) is a digital cellular standard published by the ETSI. It is the world's most popular second generation cellular system. It is used in 900 MHz, 1800 MHz and 1900 MHz bands all around the world. GSM is based on time-division multiple access (TDMA) system. Traditional GSM uses one of the 8 TDMA time-slots available, and supports full-rate speech at 13 Kbps or data at 9.6 kbps. To support higher data rate services the general packet radio service (GPRS) standard was developed based on the GSM physical layer. GPRS services can use multiple slots from the TDMA frame leading to higher data rates of up to 171 kbps.

The GSM/GPRS application would be one of many applications running on the Sandblaster concurrently. In this paper different implementation approaches for the GSM/GPRS physical layer are considered. In one approach, a single slot GSM can be run on one thread of the Sandblaster DSP. In another approach, it can be distributed across multiple threads. In the later case the Sandblaster can be powered down when not in use to save on battery power.

Handsets are converging to multimedia multi-protocol systems with multimedia and communications systems. Sandbridge Technologies has developed the multithreaded Sandblaster DSP core for such convergent devices. Depending on the real-time processing requirements of the applications, an optimal system software architecture can be designed to use the 8 threads on a single core. In a multi-core chip more threads are available and hence optimization has be performed over the whole system. GSM/GPRS is the most popular second generation cellular system in the world. This system will continue to be used for a number of years. When GSM/GPRS is integrated in such a multi-protocol system it is useful to have a flexible implementation code. Thus, based on the RTOS requirement of a given system the most appropriate implementation can be picked. Hence in this paper various ways of implementing the GSM/GPRS on the Sandblaster are presented.

A flexible implementation is also useful in saving power dissipation. On the Sandblaster individual threads can be put to sleep mode or the whole core can be powered down. Thus, by appropriately picking the implementation style of applications power dissipation can also be minimized. For example if the applications running require 50% of the MIPS and they are evenly divided among the threads the whole core can be turned off for 50% of the time, rather than individual threads being turned off at different times. This improves on power savings.

2. SANDBLASTER DSP

Sandbridge Technologies has developed the Sandblaster architecture for a convergence device [1,2]. The Sandblaster architecture supports the data types necessary for convergence devices including RISC control code, DSP, and Java.

As shown in Figure 1, the design includes a unique combination of modern techniques such as a SIMD Vector/DSP unit, a parallel reduction unit, and a RISC-based integer unit. Each processor core provides support for concurrent execution for up to eight threads of execution. All states may be saved from each individual thread and no special software support is required for interrupt processing. The machine is partitioned into a RISC-based control unit that fetches instructions from a set-associative instruction cache. Instruction space is conserved through the use of compounded instructions that are grouped into packets for execution.

The memory subsystem has been designed carefully to minimize power dissipation. The pipeline design in combination with the memory design ensures that all memories are single ported and yet the processor can sustain nearly 4 taps per cycle for a filter (the theoretical maximum) in every thread unit simultaneously. A RISC-based execution unit, depicted in the center of Figure 1, assists with control processing.

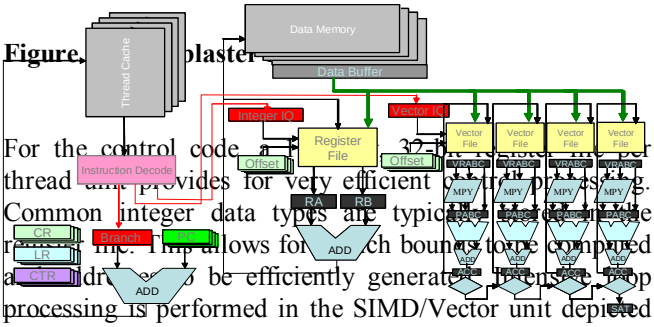


Figure. Sandblaster

For the control code a 32-bit RISC instruction set per thread unit provides for very efficient control processing. Common integer data types are typically supported in the register file. This allows for branch bounds to be computed and the instruction looped on itself until the entire vector is processed. This may be specified in as little as 64-bits.

To enable signal processing in software, the processor supports many levels of parallelism. Thread-level parallelism is supported by providing hardware support for up to 8 independent programs to be simultaneously active on a single Sandblaster core. This minimizes the latency in physical layer processing. Since many algorithms have stringent requirements on response time, multithreading is an integral technique in reducing latencies. The data-level parallelism (SIMD) is supported through the use of a Vector unit.

3. GSM/GPRS PHYSICAL LAYER

3.1 GSM/GPRS

GSM/GPRS physical layer is described in detail in the 3GPP standard documents [3]. GPRS is based on TDMA with a frame consisting of 8 slots. The frame duration is 4.615 msec and each slot is about 0.577 msec. The sample rate is 270.83 KHz with a bandwidth of 200 KHz for each channel. For traffic channels a user is allocated one or more slots (for receive and transmit). The data is modulated using GMSK modulation scheme [4]. A slot has 154.25 samples including a 26 bit training sequence and two sets of 58 sample encrypted data. The training samples are used for fine time synchronization and for channel estimation. The broadcast control channel (BCCH) from the basestation to the mobile receiver contains frequency burst (FCH) and synchronization burst (SCH). This is typically used for initial frequency and time synchronization. Channel coding of the data consists of convolutional code and BCS (block check sequence). Different code rates are available.

3.2 Receiver Processing

The receiver assumes that baseband I and Q data is provided by the RF unit. The receiver starts off by first doing a cell search to determine the strongest channels available. It steps through the all the GSM frequency bands (124 bands in the case of the European GSM between 935 to 960 MHz), and determines the 5 strongest bands. Next the receiver starts acquiring the signal starting with the strongest band. Acquisition begins with obtaining the frequency offset and frame synchronization using the BCCH channel. The FCH burst is sent periodically in the BCCH channel. It consists of a data pattern of all 0's for the whole slot. When this data is modulated by GMSK it appears as a sinusoid of frequency a quarter of the baseband T sampling i.e. 270.83/4 KHz. Such a tone can be detected by using a digital PLL or prediction based adaptive line enhancer (ALE). The offset of the tone from 270.83/4 KHz gives the frequency offset. Also, the appearance of the tone gives the location of the FCH burst. This can be considered as coarse frame synchronization. Next the receiver looks for the SCH burst which consists of a 64 bit training pattern. By correlating the received BCCH signal with a 64 bit pattern the exact location of the SCH burst is determined. With this the 8 slot frame boundary is fully acquired. During steady state the FCH and SCH bursts are used to do frequency and time tracking. The AGC operation is also performed along with the synchronization. The RF power level is initially adjusted based on the FCH power level. For AGC tracking the power level of the TCH data slots can be used.

Next the traffic channel data on the assigned slot is started. In the case of GPRS multiple slots may need to be demodulated. First the exact slot boundary is determined by doing a correlation with the 26 bit training signal in each

slot. This ensures that any timing slip is corrected for prior to demodulation of the slot. First channel estimation is performed using the 26 training bits in each slot. A 5 tap channel model is assumed. Well known least-squares method is used to optimally estimate the channel. The GMSK demodulation is achieved using a maximum likelihood sequence estimation (MLSE) algorithm. It is assumed that MSK signal is transmitted and that the Gaussian filtering of GMSK is part of the extended mobile channel. Using the viterbi algorithm the transmitted data bits are determined [4].

The remaining parts of the receiver are the differential decoder, de-ciphering and channel decoding. These are based on well known decoding methods. The transmitter functionality is well described in [3].

4. MULTITHREADING THE GSM/GPRS

4.1 Real-time Operation

The main blocks in the receiver are detailed in Fig. 2. Some of the smaller blocks are merged in the figure. The GSM/GPRS RF signal is down-converted to baseband digital samples by the RF unit. These samples of data are continuously written to buffers in the memory of the Sandblaster. The slot data handler sends all the data to the sync block until frequency and frame synchronization is achieved. After that the slot data block monitors the data in the buffers and keeps a count of the slots of the TCH and the BCCH channels, and sends the appropriate data to either the sync block or the demod unit. For example, once FCH data arrives it is sent to the sync unit and the frame synchronization process is started. If a TCH slot data is received it is sent to the demod unit and the demod process is started.

In a conventional DSP processor this kind of real-time processing would be achieved using interrupts and context switching between different processes. In a multithreaded environment the different processes are run on different threads without the overhead of context switching. All code was written in C language and compiled with the Sandbridge compiler and no assembly coding was used in this project.

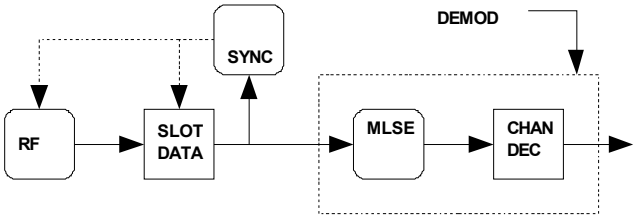
4.2 Single slot GSM/GPRS on 8 threads

In this section we consider a multithreaded implementation where the processing is spread evenly over 8 threads of the core. The threads are as shown in Table 1.

Table 1. Single slot GSM/GPRS on 8 threads

Thread Number	Functionality
1	Slot Data Handler
2	Frequency Synchronization
3	Frame synchronization
4	MLSE 1
5	MLSE 2
6	Channel Decoder
7	Tx
8	Misc.

The slot data handler synchronizes the operations of the different threads. It monitors the input data buffers and when TCH data is available it calls the MLSE. The MLSE operation is the most compute intensive block in the system. Hence this functionality is divided into two threads. The TCH data slot has two sets of 58 bit data bits with 26 training symbols in the center. The two sets of the data can be independently demodulated with MLSE, and thus forms a natural split to multithread them. The two halves of MLSE are hence processed on two threads, MLSE1 and MLSE2. The MLSE1 thread does the additional operation of channel estimation. Once MLSE1 and MLSE2 are completed the channel decoder thread is started. The frequency synchronization tracking and frame synchronization tracking are initiated by the slot data handler when the FCH and SCH bursts appear on the BCCH channel. The Tx is scheduled by the L1 control layer and hence this also initiates the Tx thread. This implementation is a trade-off



between even distribution of load on the threads, and ease of programming and minimization of thread synchronization controls. Its should be noted that if GSM/GPRS were the only application running the processor would be in power down mode for most of the time given the speed of the Sandblaster.

4.3 Four slot GPRS on 6 threads

In this section we consider a four slot GPRS running on the Sandblaster. We assume there are 4 receive slots and up to 4 transmit slots in this implementation. The threading is shown in Table 2.

Table 2: Four slot GPRS on 6 threads

Thread Number	Functionality
1	Slot Data Handler
2	Synchronization
3	Demod1 and TX1
4	Demod2 and TX2
5	Demod3 and TX3
6	Demod4 and TX4

Each of the four slots is handled on an independent thread. Here the demodulation includes MLSE and channel decoder as shown in Figure 2. The frame and frequency tracking are run on a separate thread. The slot data handler initiates the different demod threads when the corresponding slot is available.

4.4 Four slot GPRS on 6 threads alternate method

The 6 thread implementation in the previous section can be reduced to 4 threads as shown in Table 3.

Table 3: Four slot GPRS on 6 threads (alternate method)

Thread Number	Functionality
1	Slot Data Handler
2	Synchronization
3	Demod 1&2
4	Demod 3&4
5	TX1 and TX2
6	TX3 and TX4

In this case the computational load on threads 3 and 4 is higher than on the threads 1, 2, 5 or 6. Hence the time for which the core can be powered down is reduced. However, individual threads can be put to sleep when not used.

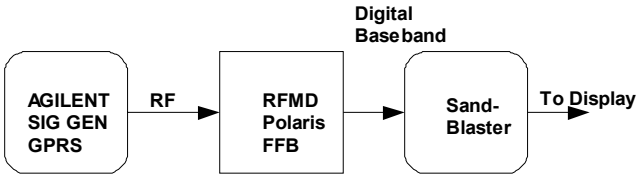
5. TESTING

5.1 Simulations to test the code

ETSI requires that the C-code for GSM be tested for different channel conditions[5]. Testing was done on channels for hilly terrain, typical urban and rural area channels. All coding schemes (CS1, CS2, CS3, CS4) were tested and they passed with significant margin. Both floating point and fixed-point codes were tested.

5.1 Hardware testing with RF signal

To test the implementation on hardware a RFMD polaris form factor board (FFB) was used as the RF front-end. This board consists of the RF2722 front-end IC, RF6001 fractional synthesizer, RF3146 power amplifier etc on a form factor board. The RF signal was generated using an Agilent signal generator (EE4438C) with a GPRS personality. The output from the RFMD Polaris board is digital baseband which is fed in the Sandblaster DSP as shown in Figure 3.



The payload data on the Agilent signal generator was different images on the different slots. The received data after demodulation in the Sandblaster is fed to an LCD screen to observe the images from the different slots. This was successfully demonstrated.

The system was further tested with a commercial T-Mobile basestation in the NY White Plains area. First the system scans the channels received over the air to determine the five highest power signals. Next the system synchronizes to the BCCH channel and completes the frequency and frame synchronization. The control data on the SCH is then demodulated to get information such as the base station identification code (BSIC), country code (CC), network code (NC) and location area code (LAC). To verify the data a commercial software called “logomanager” is used. This software works with a Nokia mobile phone to decode the control channel information. All the data was successfully verified.

6. CONCLUSIONS

A GSM/GPRS system was designed and implemented on the Sandblaster DSP and verified in hardware and software. Various ways to implement on the Sandblaster was explored. The ease of implementation and flexibility proves that Sandblaster is an efficient DSP for multi-protocol system.

7. REFERENCES

[1] John Glossner et al, "Sandblaster low power DSP", IEEE 2004 Custom Integrated Circuits Conference, 2004, pp 575-581.

[2] Sanjay Jinturkar , John Glossner, Mayan Moudgill, Erdem Hokenek, "Programming the Sandblaster Multithreaded Processor", GSPx 2003.

[3] 3GPP TS 05.01 (GPRS standard), "Physical layer on the radio path: General description," 1999.

[4] Raymond Steele, Mobile Radio Communications, Pentech Press,1994

[5] 3GPP TS 05.05 (GPRS standard), "Radio transmission and reception," 1999.