A SOFTWARE SOLUTION FOR IEEE 802.11g RECEIVER

M. Yu[1], V. Ramadurai[1], M. Senthilvelan[1], H. Ye[1], Z. Tu[1], S. Lahoti[1], J. Graziani[1], S. Agrawal[1], D. Iancu[1,2], M. Moudgill[1] and J. Glossner[1]

([1]Sandbridge Technologies Inc., Tarrytown, NY 10591, USA,
{myu,vramadurai,msenthilvelan,huaye,ztu,slahoti,jgraziani,sagrwal,diancu,mmoudgill,jglossner}@sandbridgetech.com
[2]Tampere University of Technology, Korkeakoulunkatu, FIN-33720 Tampere, Finland)

ABSTRACT

IEEE 802.11g is a widely used standard for Wireless Local Area Networks (WLAN). In this paper, we discuss a software implementation of an 802.11g WLAN baseband receiver on the Sandbridge SB3500 processor. The Sandblaster SB3500 is a compact and power-efficient system-on-chip platform tailored for wireless devices. We briefly describe the structure of SB3500. After a discussion on the structure and performance of equalizers, we introduce the details of the software defined implementation of the 802.11g implementation on the SB3500.

.

1. INTRODUCTION

IEEE 802.11g [4] is one of the most widely used standards for Wireless Local Area Networks (WLAN). The ubiquitous usage of WLAN has covered not only laptops but also smart phones, PDAs, cameras or iPods, etc. In this paper, we discuss a software solution for 802.11g WLAN baseband receiver using the Sandbridge SB3500 processor.

Our WLAN solution can coexist with other communication protocols on the same device such as smart phones or PDAs. To be capable of multi-mode communications systems, smart phones or other portable devices require flexible, small-sized and low power designs. Software Defined Radios (SDRs) offer methods to process signals digitally. They can dynamically reconfigure a receiver structure to reuse hardware, so as to update communications systems over the air as a service provider allows. The Sandbridge SB3500 processor platform is suitable for multi-mode mobile devices.

We describe the SB3500 processor first. The processor is a revolutionary system-on-chip processor. It is contains with three DSP cores and an ARM core. The DSPs have a four-threaded implementation the Sandblaster 2.0 architecture. This architecture can issue three operations per cycle, one of which can be a single instruction multiple data (SIMD) operation The SIMD unit supports 16 element vectors of 16 bits, and has with operations for complex signal processing such as Fourier transform (FFT), Viterbi and turbo decoding . The chip has proven to be flexible and powerful enough to do the baseband processing for many wireless protocols, including WCDMA, WiMAX, LTE, GPS etc.

The paper presents the system model and the major processing blocks along with the computational and system performance for exponential fading channels with additive white noise. In 802.11g frame formats are either spread-spectrum or OFDM modulation. After signal energy detection, the receiver decides on the type of modulation for the incoming frames and chooses the corresponding software branch for further processing. The main processing blocks include resampling, synchronization, equalization, and outer receiver processing.

Our main focus, in this paper, is on the equalization block as part of the end to end processing chain. There are a lot of existing works on the equalizers for WLAN. For example, [2], [5] discuss equalization for DSSS/CCK mode, and [6] discusses equalization for OFDM mode. Our paper focuses on the software implementation on the SB3500 processor. For OFDM frames, a one-step zero-forcing frequency domain equalizer is adopted. The vector unit of the SB3500 can process multiple subcarriers in parallel. For the spread-spectrum modulation, three types of time domain equalizers (LMS, LMMSE, and symbol based DFE) were compared. The choice of the equalizers affected the SW implementation on the multi-thread processor. In the paper we show the performance and complexity of each equalizer, and describe the implementation issues on the SB3500 processor in order to meet the performance requirements.

2. SANDBLASTER DSP

The SB3500® features the Sandblaster® DSP for execution of baseband in software – including physical layer. It has a programmable RF interface, with the capability to capture raw data at 240 MSPS. It includes interfaces to LCD, keypad, USIM, Smartcard, Audio codec, IrDA, plus emerging 'critical' features such as add-on memory cards, camera interface, and USB.

Sandbridge Technologies has developed the Sandblaster architecture [1] for convergence devices. As handsets are converging to multimedia multi-protocol systems, the Sandblaster architecture supports the data types necessary for convergence devices including RISC control code, DSP, and Java.

Figure 1 shows the architecture design of Sandblaster. 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 Sandblaster core provides support for concurrent execution for up to four 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.

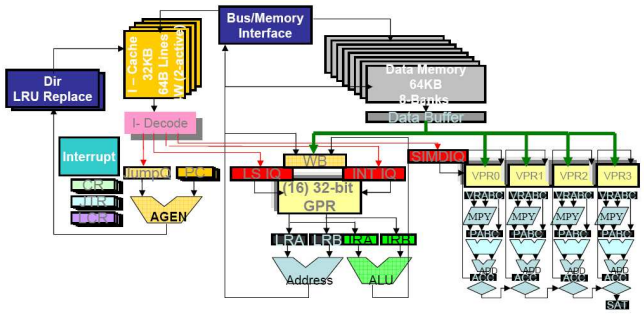

Figure 1. SB3500 DSP core

The memory subsystem has been designed carefully to minimize the 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 16 taps per cycle for a filter (the theoretical maximum) in every thread unit simultaneously. A RISC-based execution unit, shown in the center of Figure 1, assists with control processing. In the case of control code, a 16 entry, 32-bit register file per thread unit provides for very efficient control processing. Common integer data types are typically stored in register files. This allows for branch bounds to be computed and addresses to be generated efficiently. The SIMD/Vector unit depicted on the right side of Figure 1 performs intensive loop processing. Each cycle, a 16x16-bit vector may be loaded into the register file while two vectors are being multiplied, saturated, reduced (e.g. summed), and saturated again. The branch bound may also 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 multimedia processing in software, the processor supports several levels of parallelism. Thread-level parallelism is supported by providing hardware support for up to 4 independent programs to be simultaneously active on a single Sandblaster core. This reduces the latency in physical layer processing. Since many algorithms have stringent requirements on response time, multithreading is an integral technique in minimizing latencies. The data-level parallelism (SIMD) is supported through the use of a SIMD Vector unit. Additionally, the compound word instruction set provides instruction level parallelism.

3. EQUALIZATION

WLAN 802.11g frames have two modes, OFDM mode and DSSS/CCK mode. The frames of both modes consist of one preamble, one header and a data part. The preamble is for signal detection, AGC and synchronization. Channel estimation or training also uses preamble.

3.1. OFDM Mode Equalization

OFDM mode employs BPSK, QPSK, 16QAM or 64QAM modulation on subcarriers. The WLAN OFDM signal uses OFDM modulation after subcarrier mapping. The existence of a cyclic prefix (CP) in front of each OFDM symbol transforms the linear convolutional multiplication into circular convolutional multiplication (when the maximal channel delay is shorter than the length of the CP, and the synchronization eliminates inter-symbol-interference (ISI) within FFT windows.) Thus enables the implementation of frequency domain equalization. We assume the carrier frequency offset (CFO) is removed, so the signal on different subcarriers is independent from each other. A one-tap zero-forcing (ZF) equalizer is used to compensate the effect of channel fading.

Let us assume the general baseband signal model as

$$y_n = \sum_{l=0}^{L} h_l x_{n-l} + v_n, n = 0,1,2,\ldots \quad (1)$$

where x_n and y_n denote the transmitted signal and received signal respectively, h denotes the channel impulse response, and v_n the additive Gaussian noise $N(0,2\sigma^2)$.

Let Y_f and H_f denote the received signal, channel response respectively on frequency domain after FFT. The output of frequency domain ZF equalizer Z_f is expressed by

$$Z_f = Y_f / H_f \quad (2)$$

3.2. DSSS/CCK Mode Equalization

DSSS/CCK mode employs two modulation schemes. Lower data rate frames (1M and 2M bps) employ direct sequence spread spectrum. Data is differentially QPSK modulated first, then spread by a Barker code. Higher data rate frames

(5M and 11M bps) employ CCK modulation. For lower data rate frames, the good autocorrelation of the Barker code relieves the deterioration of fading channel. A simple rake receiver meets the performance requirement without the any equalizer. For higher data rate frames, an equalizer can enhance the performance remarkably.

Among various equalizers, linear equalizers have simple structures, lower complexity, and are easy to implement. Decision feedback equalizers (DFE) have higher complexity than linear equalizers, but can outperform the latter ones. We consider both linear equalizer and DFE.

3.2.1. LMMSE Equalizer

First, we consider a LMMSE Equalizer. The LMMSE [7] equalizer adopts linear minimum mean squared error criteria. Let us rewrite sequence (1) into a vector form, which is more suitable for linear filter processing, as following

$$\vec{y} = \vec{H}\vec{x} + \vec{v}$$

where $\vec{y}$ = $[y_{n+M}, y_{n+M-1}, ..., y_n]^T$, M+1 is the length of equalizer taps.

$$\vec{H} = \begin{pmatrix} h_0 & h_1 & \cdots & h_L & & 0 \\ & h_0 & h_1 & \cdots & h_L & \\ & & \ddots & & & \ddots \\ 0 & & & h_0 & h_1 & \cdots & h_L \end{pmatrix}$$

Let $\vec{w}$ denotes the coefficients of the filter. The target of MMSE filter is to minimize the mean-square error of the filter output.

$$\vec{w}_{\text{LMMSE}} = \arg\min_{w} E\{\|\vec{w}^H \vec{y} - x_n\|\}$$

The minimum value of the right part of equation is achieved given

$$\vec{w}_{\text{LMMSE}} = (\vec{H}\vec{H}^H + \sigma^2 \vec{I})^{-1} \vec{h}_a$$

Where $\vec{h}_a$ is the ath column of $\vec{H}$.

The above LMMSE requires the matrix inversion, whose computational complexity is huge when the matrix rank is high. The complexity increase cubically as the rank increases. And when the rank of the matrix is high, fixed-point matrix inversion may involve numerical problem. A simpler solution is to use iterative algorithm with low computational complexity to obtain an approximation of the LMMSE solution.

3.2.1. LMS Equalizer

The least mean-square (LMS) algorithm uses stochastic gradient algorithm to provide an approximation to the optimum MMSE solution[7]. Figure 2 shows the block diagram of a LMS equalizer.

The following shows the implementation of LMS filter. It consists of two parts.

a. Filter. The filter output is given by

$$z_n = \vec{y}^T(n)\vec{w}(n)$$

b. Adaption. The adaption part update the coefficients according to the estimation error.

1 estimation error

$$e_n = x_n - z_n$$

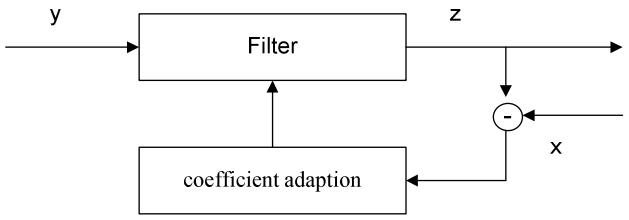

Figure 2. LMS equalizer

2. coefficient vector adaptation.

$$\vec{w}(n+1) = \vec{w}(n) - \mu e_n \vec{y}(n)$$

The coefficient vector $\vec{w}$ *(n)* can be initialized as a zero vector.

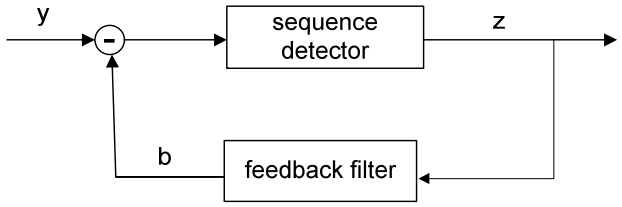

Figure 3. DFE equalizer

3.2.3. DFE Equalizer

A DFE equalizer is a nonlinear equalizer that uses previous detected decision to eliminate the ISI arising from the previous symbols on the currently processed symbols.

Figure 3 shows the block diagram of a DFE Equalizer[7]. The feedback filter reconstructs the ISI from previous the symbols by using the decision from the output of the detector. The reconstructed ISI is removed from the received signal before the input of the detector. The feedback signal can be expressed as

$$b_{k+1} = \sum_{i=1}^{\infty} x_{k-i} h_{i+1}$$

where the past decision z_n is used to as transmitted data x_n. The feedback signal is subtracted from the received signal and the result is input to the detector.

Because each CCK codeword contains 8 chips, the detector makes a decision after every 8 chips. The ISI from previous symbols is eliminated. The remaining ISI arises from the chips within the current symbol, and the ISI from the following symbols. The values of the following symbols are unknown to the detector because of causality.

Fortunately general wireless channels are close to exponential channels. In most of the time, the ISI from the following symbols does not contribute the major part of the among the total ISI. A sequence detector is employed to detect the symbol from the input sequence. The sequence detector outputs the symbol which minimize the cost function [2]

$$C = \sum_{i=0}^{7} \left| y_{k+i} - s_{k+i} e^{j\varphi_1} \right|$$

Where s_i are the 8 chips within one symbol. φ_1 is the differential phase between the previous symbol. Because the symbols are CCK modulated, the sequence detector can use fast Walsh transform (FWT) to simplify the implementation.

Figure 4 shows the performance results of the 3 types of equalizers when the data rate is 11Mbps, channel delay profile is 50ns rms delay. The x-axis and y-axis represent carrier-to-noise ratio and frame error rate (FER) respectively. The solid curve, circled curve and the squared curve represent the performance of LMMSE equalizer, LMS equalizer and DFE equalizer respectively.

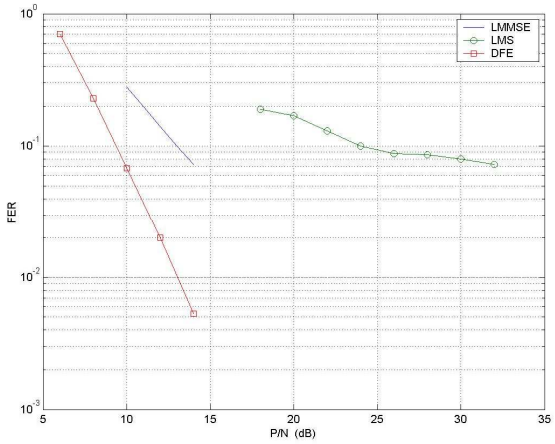

Figure 4. Performance of equalizers

From the figure, the DFE equalizer outperforms the LMMSE equalizer by 4dB at 10% FER. The DFE equalizer eliminates all the ISI from previous symbols, and uses a sequence detector. If the ISI from the following are minor, the DFE equalizer approaches the maximal likelihood detector. The LMMSE equalizer performs better than the LMS equalizer. The convergence of LMS coefficients training varies with different channel impulse response. The LMS coefficients may approach to optimal coefficients, but fluctuates. While the complexity of DFE equalizer is high, it provides the best performance among the 3 equalizers.

4. IMPLEMENTATION

Multi-threaded application has a number of advantages over a single threaded application. The SB3500 has 3 Sandblaster cores, each core has 4 threads. Assuming the Sandblaster cores are running at a clock of n MHz, then each core has n MIPs computation power, and each thread has n/4 MIPs.

To achieve real-time processing of WLAN signals, the implementation should meet two requirements. First, the training or calculation of filter coefficients is finished at least earlier than 5us after the ending of the shortest frame. Or the device fails to meet the minimum requirement for RX-to-TX turnaround time. Second, the resource available for symbol processing in one symbol's time meets the computation complexity required to process one symbol.

4.1. Implementation of OFDM Mode Equalization

For OFDM mode signal, the physical layer signal processing is pipelined into 5 threads.

Thread 1	resampling filter
Thread 2	FSO removal, FFT, channel equalizer, demapping, deinterleaver
Thread 3	Viterbi decoder
Thread 4	CRC and descrambling
Thread 5	tracking

Table 1. Thread allocation for OFDM mode

Here we focus on the thread with the equalizer only. Table 2 shows the instruction cycles need for thread 2 in each symbol time (4*us*) for 54Mbit/s data rate, and the processing power required. The deinterleaver is implemented using Scatter DMA, which operates in background.

FSO (cycles)	20
FFT (cycles)	90
equalizer (cycles)	13
demap (cycles)	100
sum (cycles)	223
Processing power (MIPS)	55.75

Table 2. Instruction cycles and processing power for OFDM equalizer

4.2. IMPLEMENTATION OF DSSS/CCK MODE EQUALIZATION

For DSSS/CCK mode, we discuss the implementations of three equalizers. The discussion assumes the highest data rate, 11Mbps.

.

4.2.1. LMS Equalizer

With an LMS equalizer, the receiver uses 2 threads at physical layer. Table 3 shows the thread allocation when processing the data

Thread 1	FSO derotation, LMS equalizer
Thread 2	CCK demodulation, bit concatenate, descramble and CRC

Table 3. Thread allocation for LMS

Table 4 shows the instruction cycles need for thread 1 in each CCK symbol, and the processing power required.

Derotation (cycles)	3
LMS equalizer (cycles)	41
Sum (cycles)	44
Processing power (MIPS)	60.5

Table 4. Instruction cycles and processing power for LMS implementation

.

4.2.2. LMMSE Equalizer

The computational complexity for LMMSE equalization of data symbols is the same as the LMS equalizer. However, the computation of the LMMSE equalizer coefficients involves a matrix inversion operation. To maintain numerical stability, we use given's rotation to inverse the matrix. The cycle estimated for the inversion is 31502 instructions on SB3500. Considering the worst case scenario, frames with short preamble, the preamble and the header last for 96us. The frame detection, automatic gain control, synchronization and channel estimation algorithms consume 39 us. This leaves 57 us for the calculation of the LMMSE equalizer coefficients. The major part of this block is a 16x16 matrix inversion which costs 30762 instructions on the SB3500. 553 MHz is required to complete this block within 57 us which is less than one core of the SB3500 [3].

.

4.2.3. DFE Equalizer

The DFE equalizer has a feedback loop from the CCK detector to the feedback filter. The detector should output the estimated signal of the current symbol before the equalizer processes the next symbol. The feedback loop made it difficult to entirely pipeline the equalizer and the detector into to two independent slots.

To simply the feedback loop, some per-frame pre-computation is implemented after channel estimation to relieve the real-time computation load on every symbol. The detector consists of fast Walsh transform and differential phase judgment. The feedback information comes from the output of the FWH. We pipeline the physical layer receiver into 2 threads.

Thread 1	Equalizer, FWH
Thread 2	Differential phase judgment. FSO derotation, bit concatenate, descramble and CRC

Table 5. Thread allocation for DFE

Table 6 shows the instruction cycles need for thread 1 in each CCK symbol time, and the processing power required.

DFE equalizer (cycles)	17
FWH (cycles)	63
Sum (cycles)	90
Processing power (MIPS)	123

Table 6. Instruction cycles and processing power for DFE Implementation

5. SUMMARY

We discussed the software solution for IEEE 802.11g baseband receiver on Sandblast SB3500 chip.

The Sandblaster® DSP is implemented as a compact and power-efficient core, and is replicated to provide a system-on-chip platform (SB3500) tailored for wireless devices. This approach scales well with successive generations of silicon process technology and provides optimum efficiency and programmability.

IEEE 802.11g receiver can be efficiently software implemented on SB3500 with low instruction cycles. The software implementation makes it convenient to for multi-mode communication system.

6. REFERENCE

[1] Sandbridge Technologies, "Sandblaster DSP Overview", www.sandbridgetech.com

[2]Y.Kim, J.Moon "CCK demodulation via symbol decision feedback equalizer," IEEE Comm Letters, Oct, 2004

[3] M. Senthilvelan, "CORDIC Instructions for Software Defined Radio," PhD Thesis, University of Wisconsin – Madison, 2009

[4] IEEE 802.11g -2003, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications : Further Higher Data Rate Extension in the 2.4 GHz Band," 2003

[5] W. Gerstacker, C. Jonietz and R. Schober, "Equalization for WLAN IEEE 802.11b," IEEE ICC04

[6] S. Boumard and A. Mammela "Channel estimation versus equalization in an OFDM WLAN system," IEEE VTC 2001-Spring

[7] J. Proakis, "Digital Communications," McGraw-Hill, 1995