COVERSHEET

Copyright Transfer Agreement: The following Copyright Transfer Agreement must be included on the cover sheet for the paper (either email or fax)—not on the paper itself.

"The authors represent that the work is original and they are the author or authors of the work, except for material quoted and referenced as text passages. Authors acknowledge that they are willing to transfer the copyright of the abstract and the completed paper to the SDR Forum for purposes of publication in the SDR Forum Conference Proceedings, on associated CD ROMS, on SDR Forum Web pages, and compilations and derivative works related to this conference, should the paper be accepted for the conference. Authors are permitted to reproduce their work, and to reuse material in whole or in part from their work; for derivative works, however, such authors may not grant third party requests for reprints or republishing."

Government employees whose work is not subject to copyright should so certify. For work performed under a U.S. Government contract, the U.S. Government has royalty-free permission to reproduce the author's work for official U.S. Government purposes.

PERFORMANCE OF SELECT BASEBAND PROCESSING LTE UE BLOCKS ON A FLEXIBLE SOFTWARE BASED BASEBAND PROCESSOR

Babak Beheshti (Sandbridge Technologies, Tarrytown, NY; b.beheshti@ieee.org); Saurabh Lahoti (Sandbridge Technologies, Tarrytown, NY); Sitij Agrawal (Sandbridge Technologies, Tarrytown, NY); John Glossner (Sandbridge Technologies, Lowell, MA)

ABSTRACT

In this paper novel approaches to implementation of several processing blocks required in the LTE standards are analyzed and benchmarked. In particular, implementation of the algorithms is assumed to be based on a flexible software based baseband processor. The blocks discussed in this paper are the FFT, the DFT and the Viterbi Decoder.

This platform therefore requires the implementation to be completely in software. This poses an important constraint, as typical hardware based implementations use very specific hardware architectural features to minimize the computational latencies. On the other hand a software implementation requires utilization of the general purpose (i.e. load/store) instructions of a processor, as well as its specialized instructions (e.g. butterfly and complex multiplications). Software based platforms have typically limited on chip (fast) memory, that may restrict storage of operating parameters required to set up an algorithm. Therefore novel techniques to maintain all time critical data in the fast memory become necessary in these implementations.

The Third Generation Partnership Project (3GPP) has been defining the Long Term Evolution (LTE) for 3G radio access. LTE project aims to ensure the continued competitiveness of the 3GPP technologies for the future LTE focuses on download rates of 100 Mbit/s, upload rates of 50 Mbit/s per 20 MHz of bandwidth, increased spectrum efficiency, and sub-5ms latency for small IP packets.

1. INTRODUCTION

Reconfigurable radio systems are radios that can change to different communication protocols as they move between different radio environments. An example would be moving from a wireless LAN 802.11b to 802.11a and then to EV-DO (Evolution Data – Optimized). Researching the development of a Reconfigurable Radio Architecture that will concurrently support multiple radio protocols over multiple frequency bands across multiple wireless networking environments is an active area of R&D in the industry. The Reconfigurable radio realizes the convergence of computing and communications by allowing a flexible communications for any handheld computing device. As more digital processing is applied to the radio system, the promise of software based digital baseband processors controlling a reconfigurable RF front end approaches reality.

Software Defined Radios (SDRs) have the potential of changing the fundamental usage model of wireless communications devices. These transceivers are often conceptually divided into two major sections: the Baseband Processing Section and the RF Front End. This division is simply a matter of convenience as the technological states of the two sections are at different stages. The baseband section which is responsible for all symbol level and bit level computations is typically implemented as reconfigurable hardware architecture or a digital signal processor (DSP).

"The SDR Forum, working in collaboration with the Institute of Electrical and Electronic Engineers (IEEE) P1900.1 group, has worked to establish a definition of SDR that provides consistency and a clear overview of the technology and its associated benefits. Simply put Software Defined Radio is defined as : "Radio in which some or all of the physical layer functions are software defined"

2. THE TARGET BASEBAND PROCESSOR

The SB3500 is the second generation of SandBlaster-based low power, high performance System on a Chip (SoC) products developed to serve the Software Defined Radio (SDR) modem applications space. As was the case for the prior generation product (the SB3011), it too is a multi-core device, however containing 3 'SBX' DSP cores (as opposed to 4), and an ARM926 processor, all interconnected by a high speed network (HSN). The ARM is intended to support protocol stacks and OS function, in addition to all the peripheral device support (such as SDIO, LCD, Camera, USB, PS/2, Smart Card, UART, DMA, AC-97/I2S, Vector Interrupt Controller, GPIO's) and external memory interfaces (Static & Dynamic Memory Controllers). The SBX DSP's each support a high rate Parallel Streaming Data (PSD) interface for the IQ baseband data, and possess the control interfaces (SPI, I2C, interrupt/timer functions, GPIO's) that are typically seen in most radio Front Ends;

they are intended to serve the primary function of Mbps wireless radio baseband data processing in software, making for a 'standards agnostic' radio platform.

Of notable improvements in this generation of the Sandblaster™ core one should cite a flexible 16-wide vector processing unit that could execute all the identified algorithms at the desired performances. The key kernels used to drive the design of this SIMD (Single Instruction Multiple Data) unit were derived from the various 3.5G and 4G standards/proposals and includes FIR, Pilot search, Descrambling, Despreading, Derotation, Complex Correlation, complex FFT (256 - 8192 points), Viterbi (constraint length 7 & 9), Turbo, and LDPC.

Each core delivers a peak of 9.6Gmacs/s once operated at 600MHz. The SB3500 therefore is capable of triple this amount or 28.8Gmacs/sec. The element-wise operations supported in this SIMD unit include common operations such as logical, shift and arithmetic operations that read 2 registers, which perform 16 short or 8 integer operations in parallel, and write the results back to a third register.

3. DISCRETE FOURIER TRANSFORM

This code is a set of functions which perform variable-size DFT's on 16 bit I/Q data. This is performed by utilizing the generalized Cooley-Tooky factorization method in which a DFT input block (which is not a power of two in size) can be decomposed into radix-2 FFT's and then reassembled to obtain the correct output. The radix-2 FFT's in this case are performed using the existing 4,8,16,32,64, and 256 RPU FFT blocks. The reassembly is performed by utilizing RPU complex multiply instructions to multiply the radix-2 FFT outputs by specialized DFT twiddle factors. To reduce code size and complexity, only branch-3 and branch-5 factorizations were utilized. For example, to perform a 360-sized DFT, the complex-valued input block would be factorized as follows:

360 DFT = 5 * 72 DFT's = 5 * (3 * 24 DFT's) =
5 * [3 * (3 * 8 FFT's)]

Each factorization step would constitute one additional depth in the factorization tree and would require multiplication with twiddle factors. The Cooley-Tooky method dictates that the DFT be evaluated as a tree structure where each depth of the tree constitutes evaluating a series of sub-DFT's. One solution is purely recursive, composed of mutually recursive functions, each of which go down one branch of the tree and spawn off different branches. This solution was attempted but resulted in higher cycles counts (due to greater function call overhead). The new, and enhanced, solution involves an iterative approach in which every sub-DFT, at every depth of the tree, is evaluated sequentially in a loop by just one function. Since the tree structure, and types of sub-DFT's, are different for each DFT size, the best way to encapsulate such a break-down for each DFT is to place the dissolution in a C-style table which each sub-DFT function is represented by a function pointer to an appropriate handle. This table allows the user to select the appropriate trade-offs between speed and memory. Since this enhanced version of the DFT implementation also allows for the dynamic, vectorized, creation of twiddle factors, the user can edit this table to decide which twiddle factors they would like the DFT functionality to dynamically create and which ones should be statically defined. The table maintains pointers to the twiddle factors for each sub-DFT and also pointers to the handler functions which would dynamically evaluate these twiddle factors.

A single DFT function is called by the user which then calls each handler, in a tight loop, to evaluate the sub-DFT's for the entire level of the DFT tree in one single uninterrupted flow. This reduces the massive function call overhead encountered in the standard implementation and also reduces code size. As a result, this particular implementation is more suited for use with applications under tighter memory and cycle controls.

For each DFT size, interleaving is done once on the input data. The interleaving performs both decimation (by 3 or by 5 depending on the factorization) as well as the necessary bit reversal (always done for any FFT) all in one. For each DFT there is a specific interleaving required depending on how the DFT is factored. The SB3500 scatter DMA is used to speed up interleaving.

In accordance with the LTE standard, the following DFT sizes were implemented and tested on the SB3500 hardware: 12, 24,36,48,60,72,96,108,120,144,180,192,216,240,288,300,324,360,384,432,480,540,576,600,648,720,768,864,900,960,972,1080,1152,1200, and.1296

The following 3500 hardware cycle counts were achieved (including the scatter DMA's). In each case, the amount of time to perform the actual DFT is less than the LTE real-time constraint for the SB3500 (~10700 thread cycles) when sampling at 30.72 Mhz. Furthermore, note that the DMA can be performed concurrently with other operations and so its latency can be completely hidden and absorbed.

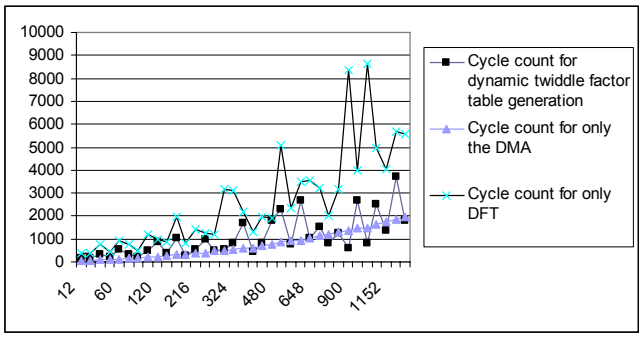

Figure 1 - DFT Cycle Counts

4. FAST FOURIER TRANSFORM

The FFT class of instructions all perform 4 complex butterflies. The instructions rfft0s, rfft1s, and rfft2s use 3 vector unit's registers in the following manner: one output (vt), one input (va), and one weights (vb). A butterfly involves a pair of complex inputs used with one complex weight to compute a pair of complex outputs. The three instructions differ in which input, output, and weight indices are used. The instruction rfft0s uses consecutive pairs of inputs and writes to consecutive pairs of outputs. It should be used for the first FFT stage. The instruction rfft1s uses alternate pairs of inputs and writes to alternate pairs of outputs. It should be used for the second FFT stage. The instruction rfft2s uses pairs of inputs with a stride of 4 and writes to pairs of outputs with a stride of 4. It should be used for the third FFT stage.
The instructions rfftls and rffths should be used for all subsequent FFT stages. They use 3 vector registers in the following manner: two input/output (vt, va), one weights (vb). For each butterfly, one complex input is used from vt and one complex input is used from va, and the complex outputs are written back to same locations in vt and va. The instruction rfftls uses the lower half of vt and va for inputs and outputs, while the instruction rffths uses the upper half of those vectors.

For an N point FFT, each stage requires N/2 complex butterflies, and there are $\log_2 N$ stages in total. Since each FFT instruction performs 4 complex butterflies, each stage requires N/8 FFT instructions. For the first 4 stages, there are no additional load/store penalties. For all subsequent stages, it takes 11 cycles per 8 FFT instructions for a total of 11N/64 cycles per stage (approximately N/6 cycles).

Therefore a good estimate for the total number of cycles for an N point FFT is (N/8)*logN for N <= 16 and (N/8)*log(16) + (N/6)*(logN - 4) = (N/2) + (N/6)*(logN - 4) = (N/6)*logN - (N/6) for N > 16.

Note that this does not include bit reversal which must be performed before the main FFT algorithm. Bit reversal takes 20 cycles per 64 complex elements shuffled, or 20N/64 for an N point FFT (approximately N/3 cycles). The final formula for cycles in an N point FFT is:
(N/8)*logN + (N/3), N <= 16
(N/6)*(logN + 1), N > 16

For the examples coded up that yields:
64 point : 75 cycles
256 point : 384 cycles
512 point: 853 cycles
1024 point: 1878 cycles
2048 point: 4096 cycles
4096 point: 8875 cycles

5. VITERBI DECODER

These functions implement different versions of a 1/3 rate, variable-length, SIHO viterbi decoder designed for a k=7, tail-biting code. Note that, in this case, none of the symbolic bits are systematic (this is based upon a 100% parity code). The optimized, pipelined, decoder is designed to be the central piece in the LTE blind decoding process for the PDCCH. The decoder is designed to perform decoding for all DCI sizes specified in the standard and achieves a performance of 6 cycles per decoder output bit. Like all SB library kernels, it is encapsulated in an easy to used functional format that accepts a sequential array of 16 bit soft input bits and produces a sequential array of unpacked hard output bits. Note that the SB3500 architecture allows for the viterbi decoder's forward ACS stage (composed exclusively of 16-way vectorized RPU instructions) to be instruction pipelined with the traceback stage of a previous decoding block (both can occupy the same instruction packet). As a result, the latency of the traceback stage can be completely hidden and absorbed.

The optimized 1/3 rate decoder can accept DCI sizes from 7 (1 + constraint length) to 96. Since the ACS stage consumes 75% of the cycles, performing it in parallel with the traceback stage would yield a further 25% reduction in cycles. The metrics given below are for the case in which the ACS stage and the traceback stage are cascaded one after the other.

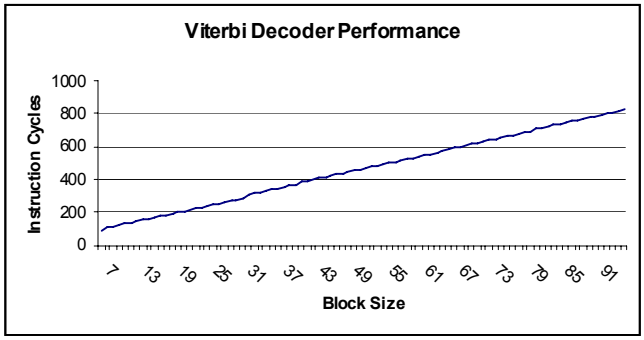

Figure 2 - Viterbi Decoder Cycle Count

6. CONCLUSION

The important conclusion derived is that in the context of an SDR implementation, various critical blocks of the LTE baseband processing can indeed be implemented entirely in software without the usage of any hardware accelerators. This important conclusion provides for a feasible solution for a multi-mode SDR supporting current wireless and cellular 2G and 3G standards, as well as emerging 4G standards.

7. REFERENCES

[1] B. Beheshti, "Analysis of a Physical Layer Wireless Communication System Implementation on an SDR Baseband Processor", proceedings of the WSEAS Conference, February, 2006, Madrid, Spain.

[2] B. Beheshti, T. Raja, "Software Defined Radio Implementation Considerations and Principles Using the Sandblaster™ SDR Baseband Processor", Proceedings of Software Defined Radio Technical Forum, 16-18 November, 2005, Anaheim, California.

[3] D. Iancu, J. Glossner, V. Kotlyar, H. Ye, M. Moudgill, and E. Hokenek, "Software Defined Global Positioning Satellite Receiver", Proceedings of the 2003 Software Defined Radio Technical Conference (SDR'03), HW-2-001, 6 pages, Orlando, Florida, 2003.

[4] J. Glossner, D. lancu, J. Lu, E. Hokenek, and M. Moudgill, "A Software Defined Communications Baseband Design", IEEE Communications Magazine, Vol. 41, No.1, pp. 120-128, Jan., 2003.

[5] J. Glossner, S. Dorward, S. Jinturkar, M. Moudgill, E. Hokenek, M. Schulte, and S. Vassiliadis, "Sandbridge Software Tools", in Proceedings of the 3rd International Worksop on Systems, Architectures, Modeling, and Simulation (SAMOS.p3), July 21-23,2003, pp. 142-147, Samos, Greece.

[6] Van Nee and Prasad, OFDM for Wireless Multimedia Communications, Artech House Publishers,

[7] ISBN 0-890006-530-6, 2000

[8] 3. T Doc #R1-060023, Cubic Metric in 3GPP-LTE, Motorola, Helsinki, January 2006

[9] 4. 3GPP TS 36.300 – v8.0.0, E-UTRA and E-UTRAN Overall Description,

[10] http://www.3gpp.org/ftp/Specs/archive/36%5Fseries/36.300/

[11] 5. 3GPP TS 36.201 – v1.0.0, LTE Physical Layer – General Description,

[12] http://www.3gpp.org/ftp/Specs/archive/36%5Fseries/36.201/

[13] 6. 3GPP TS 36.211 – v1.0.0, Physical Channels and Modulation,

[14] http://www.3gpp.org/ftp/Specs/archive/36%5Fseries/36.211/

[15] 7. 3GPP TS 36.212 – Multiplexing and Channel Coding,

[16] http://www.3gpp.org/ftp/Specs/archive/36%5Fseries/36.212/

[17] 8. 3GPP TS 36.213 – v1.0.0, Physical Layer Procedures,

[18] http://www.3gpp.org/ftp/Specs/archive/36%5Fseries/36.213/

[19] 9. 3GPP TS 36.214 – v0.1.0, Physical Layer – Measurements,

[20] http://www.3gpp.org/ftp/Specs/archive/36%5Fseries/36.214/

[21] 10. 3GPP TS 36.300 v8.0.0, E-UTRA and E-UTRAN Overall Description; Stage 2,

[22] http://www.3gpp.org/ftp/Specs/archive/36%5Fseries/36.300/