Background Data Organisation for the Low-Power Implementation in Real-Time of a Digital Audio Broadcast Receiver on a SIMD Processor

P. Op de Beeck†‡, C. Ghez†, E. Brockmeyer†, M. Miranda†, F. Catthoor†‡and G. Deconinck‡
†IMEC, Leuven, Belgium
‡ESAT Lab. Katholieke Universiteit Leuven, Leuven, Belgium

Abstract

In this work we illustrates the strong interaction between the data organisation in background memory and the data format required for sub-word level acceleration. The impact of such interaction is demonstrated on the implementation of a Digital Audio Broadcast Channel Decoder on a TriMedia processor, where data format transformations applied on the background memory data enable a substantially better exploitation of the available Single Instruction Multiple Data instructions. As a result, a factor two reduction for both execution time and data memory energy is achieved.

1 Introduction

In the near future mobile radios will be equipped with Digital Audio Broadcasting (DAB) reception. The transmission system in the DAB standard is based on an Orthogonal Frequency Division Multiplex (OFDM) transportation scheme using up to 1536 carriers [1].

Several DAB channel receivers have been reported in literature mainly implemented using either custom ASICs [2] or high-end PC-based systems [3]. However, the implementation of such advanced channel receivers using cheaper low-power multimedia instruction set processors provides a productive, flexible and cost effective alternative to these high-end solutions. To achieve that it is essential that the necessary instruction level parallelism is available via the Instruction Set Architecture (ISA) to execute the application in real-time while using lower clock rates than high-end PC-based systems. The combination of a Very Long Instruction Word (VLIW) architecture and Single Instruction Multiple Data (SIMD) ISAs provides enough degree of parallelism so as to be able to implement many current multimedia and wireless applications in real-time. However, this is only attainable on condition that the format of the data in background memory and the one required by the sub-word level units matches, given the different requirements for the write and read related operations of the shared background data this is far from obvious.

Compiler support for SIMD instructions is largely lacking in most existing architectures, and it typically requires the designer to manually insert calls to optimised libraries at the source code level. Still, even if this is done so, an overhead occurs when the data format required for the SIMD operation does not match that one chosen for background data storage. The consequence is that the potentially obtainable speed-up provided by the SIMD operation is hidden by the cycle overhead of the extra data formating instructions required (e.g. SIMD (un)pack operations). These extra formatting operations are necessary to combine/split the data words before/after these are processed by the SIMD unit. Our goal is to show how that overhead can be avoided by adapting the data storage organisation in a processor and memory platform dependent way.

The illustration of such interaction is the focus of this paper. For illustration purposes, we have chosen the TriMedia TM1300 [9] because of its combined VLIW and SIMD nature of its ISA. However, our observations are applicable to any architecture supporting sub-word level processing in its instruction set.

2 Related Work

In [4] the data format decisions (e.g., data (un)packing) is tackled during the code selection phase. Only the data needed together is merged together in background memory. However, no prior code transformations are explored to create more freedom. In [7] memory accesses are coalesced in order to more efficiently use a processors memory system and recently [5] has proposed a method that horizontally orders the scalar data during address assignment to improve program performance in SIMD processors however this .after the scheduling which limits the freedom for background data merging.

Also in the custom processor domain data merging has been explored [6] for dynamically allocated record types. However, the additional freedom of manifestly specified arrays is not looked at and the approach does not consider constraints coming from the (SIMD) operation format.

3 Basic group structuring transformations for SIMD data format organisation

A crucial module in the DAB is the OFDM decoder that at the receiver side is implemented as a forward complex FFT. The starting point of our experiments has been an

OFDM kernel which has been optimised in a Platform Independent manner [10].

The regular structure of the FFT butterfly (Figure 1a) makes it a good candidate for sub-word level acceleration. The mapping onto the TriMedia instruction set is shown in Figure 1b. The key feature is to re-interpret the content of a register as two sub-words. In this way the total amount of word-level arithmetic operations has been reduced by more than a factor 2. However, many extra packing instructions are introduced to correctly feed the SIMD operations (see Figure 1). This penalty in extra cycles can be avoided when data is already merged in the background memory by applying the so called Basic Group Structuring transformation in the source code.

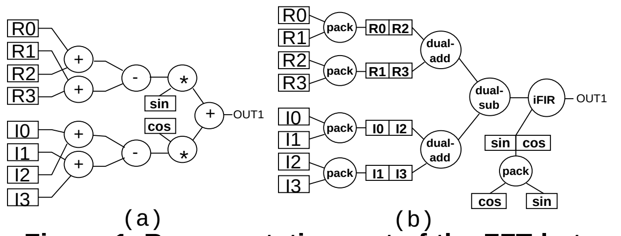

Figure 1. Representative part of the FFT butterfly before (a) and after (b) SIMD

Basic group structuring (BGS) is a transformation [11, 6] that horizontally merges the elements of different arrays in one Basic Group (BG) array. Usually these BG arrays contains elements that are processed together. BGS can have a large impact on data memory, power, size and bandwidth.

4 Experimental framework and results

This section experimentally verifies the need for BGS transformations. For that purpose we have obtained three versions corresponding to the platform independent optimised code [10]; (2) a transformed code where SIMD operations have been used; (3) a BGS transformed code where both, the Real and Imaginary input arrays have been merged into one BG and, the `Sine/Cosine` coefficient arrays have been merged also into another one BG both in background memory. This last version has been also complemented using SIMD operations, i.e., `dual_add, dual_sub` and Trimedia's `iFIR16` for the multiply/add coefficient operation (see Figure 1). This last version effectively reduces the number of load/store operations to these new BG arrays by two while enabling their processing within the same SIMD instruction.

In our experiments we have mainly focused on the data memory energy consumption and the overall execution time. The energy model used for the data memory is defined as:

$$E_{tot} = N_{hits} * E_{cache}^{tag+data} + N_{miss} * E_{cache}^{tag} + (N_{miss} + N_{cb}) * E_{SRAM}$$

with N_{hits}, N_{miss} and N_{cb} the number of cache hits, cache misses and copy-backs respectively. $E_{cache}^{tag+data}$ is the energy per access to the cache activating both tag and data lines. E_{SRAM} is the energy per access to main memory. We use the Cacti model [8] with a .35μm SRAM technology to estimate the energy per access.

Table 1 shows the energy consumed for one second of audio stream, the total execution time, the number of load/store operations and the number of misses for all three code versions. This table clearly shows that the BGS+SIMD alternative outperforms the SIMD-only version. This is mostly due to a decrease in the number of data accesses but also due to less packing operations needed for the SIMD processing. In other words, the potential gain that one would expect (a factor 2) from using SIMD instructions was actually hidden by a (un)pack instruction overhead in the original code.

Implementation	Energy Mem.(mJ)	Exec. time(sec)	# ld/st (10^6)	# misses (10^6)
PI optimised	67.2	0.413	43.55	0.60
SIMD-only	67.1	0.411	43.48	0.59
BGS+SIMD	28.5	0.248	18.20	0.65

Table 1. Impact of BGS on SIMD acceleration

5 Conclusion

In this paper we have demonstrated on an OFDM kernel that using SIMD instructions requires the data to be merged in background memory. If this Basic Group Structuring transformation is omitted the sub-word level acceleration capabilities provided by the architecture will be sub-optimally exploited due to a pack instruction overhead.

References

[1] European Telecommunication Standard ETS 300 401, “Radio broadcasting systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers”, *RE/JTC-00DAB-4*, May 1997, Second Edition

[2] J. Huisken, F. van de Laar, M. Bekooij, G. Gielis, P. Gruijters, F. Welten, “A Power-Efficient Single-Chip OFDM Demodulator and Channel Decoder for Multimedia Broadcasting”, *IEEE Journal of Solid-State Circuits*, Vol. 33, nr 11, pp.1793-1798, Nov. 1998.

[3] D.Nathan, B.Sputh, O.Faust, C.B.Koon, ”Design and Features of an Intelligent PC-based DAB Receiver” *IEEE Trans. Consumer Electronics*, Vol. 48, nr 2, pp.322-328, May, 2002.

[4] R. Leupers and S. Bashford, “Graph based Code Selection Techniques for Embedded Processors”, in ACM Design Automation of Electronic Systems, vol. 5, no. 4, pp. 794-814, October 2000.

[5] M. Lorenz, D. Kottman, S. Bashford, R. Leupers and P. Marwedel, “Optimized Address Assignment for DSPs with SIMD Memory Accesses”, in Proceedings of ASP-DAC, January 2001.

[6] P. Ellervee, M. Miranda, F. Catthoor and A. Hemani, “System-level Data-format Exploration for Dynamically Allocated Data Structures”, in IEEE Trans. on Computer-Aided Design, vol. 20, no. 12, pp. 1469-1472, December 2001.

[7] J.W. Davidson and S. Jinturkar, “Memory Access Coalescing: A Technique for Eliminating Redundant Memory Accesses”, in Proceedings of PLDI, June 1994, pp. 186-195.

[8] http://research.compaq.com/wrl/people/jouppi/CACTI.html

[9] http://www.semiconductors.philips.com/trimedia/products/media_proc_ic/

[10] P. Op de Beeck, C. Ghez, E. Brockmeyer, M. Miranda, F. Catthoor, G. Deconinck ”Low-Power Implementation of an OFDM based Channel Receiver in Real-Time Using a Low-end Media Processor” in Proceedings IEEE CAS Workshop on Wireless Communications and Networking Pasadena CA, Sept. 2002 .

[11] F.Catthoor, K.Danckaert, C.Kulkarni, T.Omnes, *Data transfer and storage architecture issues and exploration in multimedia processors*, book chapter in “Programmable Digital Signal Processors: Architecture, Programming, and Applications” (ed. Y.H.Yu), Marcel Dekker, Inc., New York, 2001.