A Combined Hardware/Software Optimization Framework for Signal Representation and Recognition

Melina Demertzi[1], Pedro Diniz[2], Mary W. Hall[1], Anna C. Gilbert[3], and Yi Wang[3,⋆]

[1] USC/Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292 USA
[2] Instituto Superior Técnico
Technical University of Lisbon
Tagus Park, 2780 Porto Salvo
PORTUGAL
[3] The University of Michigan
Ann Arbor, MI 48109 USA

Abstract. This paper describes a signal recognition system that is jointly optimized from mathematical representation, algorithm design and final implementation. The goal is to exploit signal properties to jointly optimize a computation, beginning with first principles (mathematical representation) and completed with implementation. We use a BestBasis algorithm to search a large collection of orthogonal transforms derived from the Walsh-Hadamard transform to find a series of transforms which best discriminate among signal classes.The implementation exploits the structure of these matrices to compress the matrix representation, and in the process of multiplying the signal by the transform, reuse the results of prior computation and parallelize the implementation in hardware. Through this joint optimization, this dynamic, data-driven system is able to yield much more highly optimized results than if the optimizations were performed statically and in isolation. We provide results taken from applying this system to real input signals of spoken digits, and perform the initial analyses to demonstrate the properties of the transform matrices lead to optimized solutions.

1 Introduction

Signal and speech recognition require precision and robustness in the detection process. Not only do different peoples' vocal patterns have different spectral representations, but also within an utterance, phonemes have a variety of spectral signatures. All of these spectral patterns change over time, possibly even abruptly with a single utterance. For these reasons, speech recognition is a challenging task. When the desired detection system includes hard constraints on

⋆ This work supported by NSF CNS DDDAS-SMRP: 0540407.

hardware and software resources, as in embedded platforms, using compute-intensive recognition systems based solely upon the spectral analysis from the Fast-Fourier Transform (FFT) might exceed available compute resources or power/energy budgets.

In this paper, we describe an approach to speech recognition that is highly optimized according to the expected signals or phonemes, through a *joint optimization of mathematical representation, and hardware or software implementation.* The overall goal of the project is to exploit signal properties in signal processing applications to jointly optimize a computation, beginning from first principles with the mathematical representation, and carried through to the implementation in hardware and software. It is data-driven, in that the signal properties guide the optimization strategy, and it is dynamic, in that the algorithm execution will ultimately be tailored to the input signal. The optimization strategy begins with the mathematical derivation of a transform most appropriate for a training set of phonemes. From this mathematical representation, we exploit properties of the transform structure to significantly reduce the computations that are performed and the storage and memory bandwidth requirements, as compared to FFTs or straightforward implementations of multiplying the signal by the transform. At the implementation level, we exploit the ability to derive customized hardware implementations using field-programmable gate-arrays (FPGAs) and/or the corresponding software implementation.

Overall, the richness of the design space of mathematical transformations and hardware/software implementations makes the development of good solutions accessible to a handful of highly trained specialists only. A common approach relies on optimizing the individual components of the system. We believe a better solution lies in starting with a high-level representation of the problem that can be optimized from first principles. Thus, while this paper focuses on speech recognition of digits, this work can be thought of as a starting point for a more general methodology for deriving highly optimized signal recognition systems. Similar strategies are used in optimizing signal processing in SPIRAL [5,3], which optimizes linear transforms from high-level expression of signal processing formulas, but this paper is looking at a different transform and specifically targeting a hardware implementation.

To illustrate the concept of joint mathematical, hardware or software optimization for voice recognition, we consider the problem of recognizing spoken digits “0” through “9”, where “0” is represented by both “oh” and “zero”. A common approach to this digit recognition problem relies on using training samples to derive custom transform matrices associated with one or more digits that are then used to distinguish signals from other digits, as shown in Figure 1. In this figure, we see a process where a training set of signals representing digits are analyzed offline to derive, in this case, a transform associated with each spoken digit. The BestBasis algorithm derives a series of transforms which will best discriminate among the digit classes. Each transform encodes a choice of feature vectors or set for the signal classes. In the on-line processing phase, these

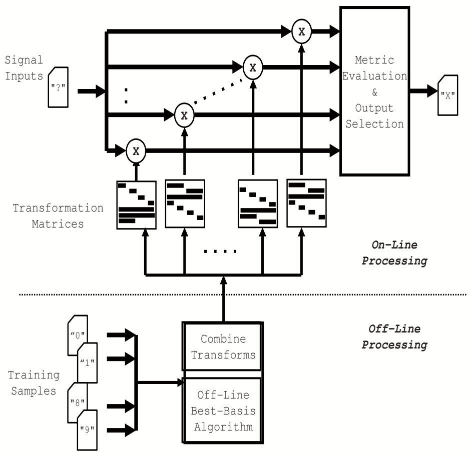

Fig. 1. Overview of speech recognition approach

transforms are applied to the input signals, followed by the metric evaluation and output selection step of a classifier to determine which digit has been spoken.

This paper is organized as follows. In the next section we describe the mathematical foundation of our specific domain and in Section 3 we describe a set of optimization opportunities in the context of the hardware implementation of a specific set of transformations. Section 4 describes some preliminary results, followed by a conclusion.

2 Mathematical Foundations

2.1 Walsh Wavelet Packets

The Walsh-Hadamard transform H_n of size $2^n \times 2^n$ is an orthogonal transform closely related to the Fourier transform. Because the Walsh-Hadamard transform is an orthonormal basis, each row of the matrix H_n corresponds to a basis vector. Let W_n^ℓ denote the ℓth row vector of the Hadamard matrix H_n. We use these vectors to build a redundant or overcomplete collection of vectors of size $2^L \geq 2^n$. To form a Walsh wavelet packet vector, we choose some W_n^ℓ of length $2^n \leq 2^L$ and we "insert" it in a vector of zeros (of length 2^L) at position $2^n k$ for some integer $k = 0, \ldots, 2^{L-n} - 1$. The collection of all such vectors $W_{n,k}^\ell$ for $n = 0, \ldots, L-1$, $k = 0, \ldots, 2^{L-n} - 1$, and $\ell = 0, \ldots, 2^n - 1$ is the full collection of Walsh wavelet packets of length 2^L. This collection defines a matrix with dimensions $L2^L \times 2^L$. If we apply the matrix to a vector, we obtain more wavelet packet coefficients than points in the vector.

Because the collection of Walsh wavelet packets is overcomplete, there are multiple ways in which to represent a vector as a linear combination of wavelet packets. This collection even includes an exponential-sized family of orthonormal bases, which means that there are exponentially many different transforms to choose from, including those with good discriminatory quality, good compressive ability, etc. We can choose our basis in which to represent our vector depending on different design criteria.

In Figure 2, we illustrate two different matrices which instantiate two different orthonormal transforms of a vector of length $2^L = 8$. Observe that the matrices have different sparsity structure; that is, the matrix on the left has many fewer nonzero entries than the matrix on the right. The notation for the Walsh wavelet

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \qquad \begin{bmatrix} \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \\ \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} \end{bmatrix}$$

Fig. 2. Two matrices which instantiate two different orthonormal Walsh wavelet packet transformations of a vector of length $2^L = 8$

packet vectors encodes the structure of the nonzero blocks we see in the rows of the matrices in Figure 2. We denote each vector by $W_{n,k}^{\ell}$ where n determines the extent or width of the non-zero block, k determines which column the block begins in (column $2^n k$), and ℓ determines the sign pattern on the block. Because we have an orthonormal transform, each pattern is normalized by $2^{-n/2}$. Observe that both the block length and the initial column are powers of 2.

2.2 Best-Basis Algorithm

Although the collection of orthonormal transforms is exponentially large, the BestBasis algorithm searches this collection efficiently because of the implicit binary structure of this collection. We obtain the Walsh wavelet packet coefficients of the signal by recursively applying the H_1 matrix to the signal. This recursive application generates a binary tree of wavelet packet coefficients.

We index the nodes B_n^{ℓ} of the binary tree with n for the scale or length of the Walsh wavelet packet and ℓ for the sign pattern or frequency. Within each node B_n^{ℓ}, we have the coefficients corresponding to the vectors $W_{n,k}^{\ell}$. An orthonormal basis in the exponential family is a maximal antichain in the binary tree and there are $O(2^L)$ such bases. This decomposition allows us to formulate a dynamic programming algorithm to efficiently optimize over all admissible bases. See [1] for the details of the BestBasis algorithm. In our case, the optimization criterion is the discriminability of the basis and we refer to [4] for details of the discrimination criterion.

3 Exploiting Transformation Structure in Hardware

As described in section 2, the basic Walsh-Hadamard transform and its derived Walsh wavelet packet transforms have a very specific structure that enables various hardware-oriented improvements, when computing the dot-product of a transformation matrix with an input signal vector. We now describe some of these opportunities.

3.1 Compressed Representation

The transformation matrices described in the previous section have a structure that lends itself to a compressed data representation. Each row of the matrix has a single non-zero block that can be represented as a coefficient that is a specific power-of-two and a non-zero block of only $+1$ and -1 coefficients.

Consider a 4096×4096 matrix, which in 32-bit single precision would be represented in its binary form as a more than 60 Mbyte object. Given the above structure, we can greatly compress this representation to reduce bandwidth and storage requirements of the matrix. Each row can be represented with the following set of information:

1. n, where 2^n is the length of the block, and $2^{-n/2}$ represents the coefficient of the block.
2. $index$, the position in the row representing the start of the block.
3. $b(index, index + 2^{n-1})$ a set of 2^n bits, with each bit representing the sign of the corresponding element in the row (0 for 1, 1 for -1).

In the worst case, where the entire block is non-zero, each row of the matrix takes 12 bits to represent n (limited to 4096), 12 bits to represent $index$, and 4096 bits to represent b, or 515 bytes per row and no more than 2 MBytes for the entire matrix. We see in the next section that most rows of the matrices contain many zeros, so we can compress the matrices much more than this.

3.2 Optimized Implementation for a Row

To compute a single coefficient, we multiply a single row of the matrix with the input vector. Because of the structure of any single given row, we need only perform a multiplication by b, the bit pattern, and a floating point number $2^{-n/2}$. A single multiplication by a coefficient $2^{-n/2}$ is done relatively inexpensively in floating-point representation. Furthermore, the bit patterns can be implemented efficiently as well. For example, the specific non-zero block $b(0, 4)$ can be implemented fairly efficiently using a custom three-adder hardware module in two steps. During the first step, all the values are added using three adders and making use of a arithmetic negation unit. In a second step the interim result is multiplied by the $2^{-4/2} = 2^{-2}$ value. Figure 3 (left) illustrates this computation using a dedicated or custom hardware implementation.

An alternative implementation that exploits the fact that the block $b(0, 4)$ can be decomposed into two sub-blocks with the same coefficients but with opposite sign value is illustrated in Figure 3 (right). Here the computation is done in three steps but only two hardware adders are required. During the first step the values corresponding to the two $+1$ coefficients are added and saved in a register. In a second step the same adders are used to add the values corresponding to

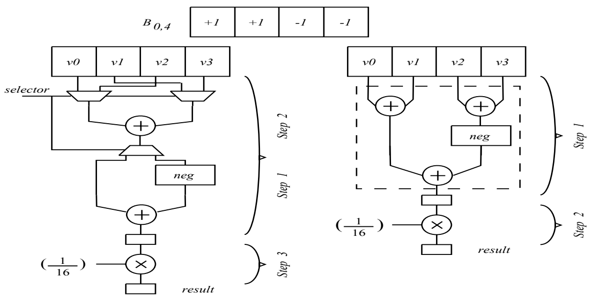

Fig. 3. A Custom Dot-Product Unit for $b(0, 4)$

the two -1 coefficients. The ouput is complemented to account for the negative sign of the coefficient. Finally in a third step, two partial results are added and multiplied by the 2^{-2} value. We can see that a faster implementation is possible if we double the number of adders and perform all four additions in a single step. A clear space-time trade-off can also be exploited in the implementation of these operations.

We make a final observation that the multiplication by the power of two can frequently be eliminated (depending on whether n is even or odd) and replaced with a shift operation. We will see in the next section that the number of distinct powers of two is small so it is straightforward to anticipate likely values for this shift prior to hardware design.

3.3 Computation Reuse

Because of the specific representation of non-zero blocks, an implementation can share partial results across rows. If two rows have non-zero blocks that overlap across the columns and there is a subset of the unitary coefficients with the same or opposite sign, then the partial sums when computing the matrix vector product can be reused. Figure 4 depicts such opportunity for reuse. These opportunities require that the actual hardware design be more sophisticated as it needs to track opportunities for partial reuse, and index and save partial results in internal memory or registers.

In addition to the reuse opportunities within one specific transform (a single matrix-vector multiplication), there are also opportunities for data and computation reuse and parallel execution when we apply multiple transforms of the same input signal vector.

3.4 Adaptive Transformation Application

We have assumed that the digit recognition procedure is a simple sequential decision process where at each step we use a transformation built to identify a specific digit. This is a sequential process that can be improved by organizing the decision tree as a binary tree or even as a single node from which emanate multiple branches corresponding to the decision of each digit. The problem of a

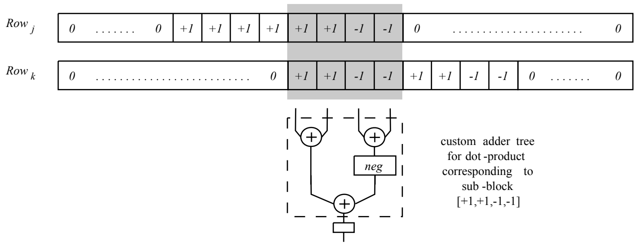

Fig. 4. Partial Computation Reuse Example

recognition in a single step or as a single transformation is that the reliability of the transformation in disambiguating all the digits might be low. On the other hand testing individually each digit is time consuming. This suggests an adaptive strategy where the implementation would start rather optimistically with a single transformation. This would allow it to identify a set of possible candidates for the input digit, say d of the 10 digits. Next the implementation would use a specific precomputed transformation for those d or simply iterate sequential over individual transformations for those d digits to clearly identify them.

4 Preliminary Results

We now present preliminary results derived from real transformation matrices obtained from training samples. These were clean spoken digits from the training portion of the TIDIGITS corpus [2]. There are three male and three female speakers. Each speaker utters the digits “zero” through “nine” and “oh” in isolation. To derive the transformation matrices, we perform the BestBasis algorithm from Section 2. Each sample included a 4096 timed sample thus leading to matrices of 4096×4096 values and to input vectors that are 4096 elements long.

As expected, the number of non-zero values for each of the transformation matrices is fairly low. Overall, for the male subjects the number of non-zeros is 15% whereas for the females this number is only 10%. For the male subjects the number of non-zeros is higher for the higher-valued digits whereas for the female subjects the reverse is true.

For this same set of matrices we determine the size of the non-zero blocks across all rows of the various matrices. The block sizes 64, 1024 and 2048 occur in every matrix, whereas other matrices may lack one of the other block sizes. For brevity, we omit these histograms.

We have developed a 4×4 custom hardware adder-tree unit depicted in Figure 3 using Xilinx ISE 7.1i tool set and targeting the Virtex-II-Pro (xc4vfx140) device. This design uses $1,714$ slices (2% of the device capacity) and attains a clock rate of 199.3 MHz. This unit is fully pipelined with an initiation interval of 12 clock cycles and overall latency of 30 clock cycles. The multiplier introduces an additional 5 clock cycle latency as it corresponds to a set of shift and normalization operations.

We then compare the performance of this custom unit with a base implementation with one single-precision adder and multiplication unit for a sample set of input signals and matrices from both female and male subjects for a wide range of input digits. For these input signals and input matrices we simulate the performance of the matrix-vector dot-product calculation and compare the performance relative to a hardware implementation using a single adder and multiplication unit. The relative performance results reveal that on average the speedup using this custom unit across all matrices from female subjects is 3.88 and is 3.91 for the male subjects respectively. We attribute this slight difference to the fact that for male subjects there are several matrices that have blocks of unit size which do not require an addition operation and have trivial multiplication. Irrespective of this minor performance difference, the speedups are nearly ideal for a custom unit performing four concurrent addition operations.

5 Conclusion

In this work, we explore the opportunities for data-driven signal processing optimization using transforms derived from the Walsh-Hadamard transform and the BestBasis selection algorithm. We are currently exploring a wide range of optimization opportunities for these transforms and the specific problem of digit recognition to highlight the opportunities for cross domain joint optimization between the mathematical, software and hardware implementation domains. This approach is data-driven, in that we use the signal properties to guide optimization, and dynamic, in that we will ultimately use features of incoming signals to optimize execution. We provide preliminary results taken from applying this system to real input signals of spoken digits, and perform the initial analyses to demonstrate that exploiting the properties of the transform matrices leads to optimized solutions. The preliminary evidence leads us to believe that this general concept is also applicable to other signal domains, not just digit recognition, particularly in resource-constrained environments.

References

1. R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 38(2):713–718, 1992.
2. D. Ellis. Sound examples `http://www.ee.columbia.edu/~dpwe/sounds/tidigits`.
3. M. Puschel, J. Moura, J. Johnson, D. Padua, M. Veloso, B. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. W. Johnson, and N. Rizzolo. Spiral: Code generation for dsp transforms. *Proceedings of the IEEE special issue on Program Generation, Optimization, and Adaptation*, 93(2):232–275, 2005.
4. N. Saito and R. R. Coifman. Local discriminant bases. *Mathematical Imaging: Wavelet Applications in Signal and Image Processing, Proc. SPIE*, 2303, 1994.
5. J. Xiong, J. Johnson, R. Johnson, and D. Padua. Spl: A language and compiler for dsp algorithms. In *Proc. of the ACM 2001 Conference on Programming Language Design and Implementation*, 2001.