EDDA — an Algorithm for Interactive Design of Hardware-efficient FIR Filters

Per Persson
Dept. of Electroscience
Lund University
Box 118, SE-221 00 Lund, Sweden
Email: per.persson@es.lth.se

Abstract— In this paper an optimization algorithm applicable to, but not limited to, the problem of designing digital filters with coefficients implemented as a sums of signed power-of-two terms is presented. The algorithm can handle general constraints which are imposed in a soft manner through continuous parameters allowing for a trade-off between frequency response error and realization complexity.

The algorithm is based on analogies from statistical mechanics and utilizes an entropy-directed deterministic annealing process to find a near-optimal solution in a time short enough to allow interactive parameter tuning.

I. INTRODUCTION

The problem of finding suitable filter coefficients given a limited wordlength is far from trivial and the problem gets worse when a simplified realization through the use of coefficients with particular properties are desired. Coefficients made up of a minimum number of signed power-of-two (SPT) terms are desirable but difficult to design, see e.g. [1], [2], [3], [4] for some recent approaches.

The coefficient selection can be considered a search for optimality over a state space consisting of \(3^L\) possible outcomes of \(L\) variables with values from the set \(\{1, 0, -1\}\), where \(B\) is the maximum coefficient wordlength and \(N\) is the maximum number of coefficients.

While many people choose to perform an intelligent search of the quantized coefficient space in the vicinity of the continuous coefficient values [1], [5], [6] it is possible to do a wider search given an algorithm that is fast enough.

This paper is laid out as follows. In Section II the method for stating a filter design problem as a combinatorial optimization problem is outlined and in Section III the entropy-directed deterministic annealing (EDDA) algorithm is presented. Section IV contains a performance analysis and summarizing comments are given in Section V.

II. DIGITAL FILTER DESIGN AS A COMBINATORIAL OPTIMIZATION PROBLEM

In order to facilitate simultaneous optimization of discrete filter coefficients and hardware realization aspects the specification is encoded as a state vector, belonging to the set \(\mathcal{S}\)

\[
\mathcal{S} = \{s|s = \{s_i\}, s_i \in \{-1, 0, 1\}, i = 1, \ldots, L\}
\]

and an associated cost \(f(s)\) with a global minimum corresponding to the optimal solution. The variable \(s_i\) is interpreted as \(j^{th}\) term of the \(n^{th}\) coefficient by the relation \(i = nB + j\).

Since we may be dealing with several aspects of the design at the same time, it is natural to consider a cost

\[
f(s) = f_0(s) + \alpha_1 f_1(s) + \alpha_2 f_2(s) + \ldots
\]

where each term is associated with a certain property.

The frequency response is given by \(H(\omega) = s^T \psi(\omega)\), where \(\psi(\omega)\) is the Kronecker product of \(\phi(\omega)\), a vector of basis functions \(\phi_n(\omega) = e^{-j\omega n}, n = 0, \ldots, N - 1\), and a vector of powers-of-two

\[
\phi(\omega) = \phi(\omega) \otimes [2^{-1} \ldots 2^{-B}]^T.
\]

Given a desired frequency response \(H_d(\omega)\) we can define the filter design cost, \(f_0\), as the integrated squared error

\[
f_0 = \int_\Omega w^2(\omega)|H(\omega) - H_d(\omega)|^2 d\omega
\]

which measures the weighted deviation from the desired response over the design bands, \(\Omega\), with weighting factor \(w(\omega)\). The solution to (4) in terms of infinite precision coefficients, denoted \(f_{LS}\), is assumed to be known.

We can rewrite (4) as a quadratic form in \(s\)

\[
f_0(s) = s^T R s - 2p^T s + g.
\]

using the definitions

\[
R = \int_\Omega w^2(\omega)\psi(\omega)\psi^T(\omega) d\omega
\]

\[
p = \int_\Omega w^2(\omega)\psi(\omega)H_d(\omega) d\omega
\]

\[
g = \int_\Omega w^2(\omega)H_d^2(\omega) d\omega.
\]

To minimize the total number of non-zero SPT terms assigned to the filter, a penalty term, \(f_1(s)\), proportional to \(s^T s\) is added to the cost, rendering a total cost of

\[
f(s) = \frac{s^T R s - 2p^T s + g}{\alpha} + \frac{f_{LS}}{L} s^T s
\]

where \(f_{LS}/L\) is a normalizing factor while \(\alpha \geq 0\) is a parameter determining the trade-off between filter design error and total number of non-zero SPT terms.
III. A Fast Deterministic Annealing Algorithm

The well known simulated annealing (SA) algorithm [7] is known to find good solutions to hard combinatorial problems, given enough time. The mean field annealing (MFA) algorithm [8], [9] was introduced as an alternative, finding solutions comparable in quality but in a fraction of the time.

Mean field annealing applies a probabilistic interpretation to the $L$ discrete state variables, mapping them to a new set of $L$ continuous variables and the solution is obtained as a stationary point of the mean field equations[8], [10]. During the course of optimization an external control parameter, $c \geq 0$, governs the search in such a way that the overall features are assessed first ($c \gg 0$) and more and more detail is gradually explored ($c \to 0$). A sigmoid-like function, emerging from the probabilistic interpretation, gradually forces the continuous variables to become either 0, 1 or -1 in the end.

The core of the algorithm is an equation [9], [11]

$$v_i = \sum_{s_i}^{s_i} \frac{s_i e^{-f(s_i, v_i)/c}}{\sum_{s_i} e^{-f(s_i, v_i)/c}}$$

used to track the optimum $v_i$, corresponding to a fixed point $v^*$, at every value of $c$, where $v_i = E[s_i]$ is real valued in the range $v_i \in [-1, 1]$ and $\bar{v}_i$ denotes a state vector $\{v_j\}_{j \neq i}$ of length $L - 1$. It is assumed that (10), from a starting solution $v$ close to $v^*$ for a given $c$, allows us to iteratively find an estimate of $E[s_i]$ and track the sequence of solutions by carefully lowering (annealing) $c$ in a sequence $c_{k+1} < c_k$, where $k = 0, \ldots, K - 1$.

Considering that the feasible solutions to the optimization problem consists of discrete variables, a necessary requirement on $v_i$ in the limit as $c \to 0$ is that they approach $s_i$. It is indeed easy to show that $\lim_{c \to 0} v_i \in \{-1, 0, 1\}$ [12].

By applying the same reasoning and approximation as in the derivation of (10) it can be shown [12] that it is possible to estimate the conditional entropy $H(s_i|\bar{v}_i)$ as

$$H(s_i|\bar{v}_i) \approx -\sum_{s_i}^{s_i} \frac{e^{-f(s_i, \bar{v}_i)/c}}{\sum_{s_i} e^{-f(s_i, \bar{v}_i)/c}} \log_2 \sum_{s_i}^{s_i} \frac{e^{-f(s_i, \bar{v}_i)/c}}{\sum_{s_i} e^{-f(s_i, \bar{v}_i)/c}}$$

(11)

In the following, $H(s_i|\bar{v}_i)$ will be used to denote the approximate conditional entropy as given by (11). Furthermore, the entropy of the system can subsequently be estimated by

$$H(s) \approx H(v) \equiv \sum_{i} H(s_i|\bar{v}_i).$$

(12)

The estimates of entropy and conditional entropy can be used to improve the algorithm in two ways. Since the entropy goes to zero when the algorithm approaches the optimal (under the approximation) state, it can be used as a stopping criterion. Furthermore, the conditional entropy estimate can be used to reduce the number of cost function evaluations by detecting when a certain state variable no longer needs to be updated.

A. Initial State and Annealing Schedule

It is possible to derive an initial estimate of $v$ by considering (10) for large values of $c$, and $c_0$ should simply be large enough that $v_i \approx 0$. A suitable annealing schedule, adopted from SA, is to let $c_{k+1} = \lambda c_k$, with $0 < \lambda < 1$ commonly chosen in the range $0.8 - 0.99$ [13].

From the previous discussion of entropy, a reliable stopping criterion can be defined as

$$H(v) = \sum_{i} H(s_i|\bar{v}_i) < \varepsilon_{\text{stop}}.$$

(13)

which has the nice property that an $\varepsilon_{\text{stop}}$ value of $\log_2 3$ corresponds to the entropy of a single undecided variable.

B. Determining Stationarity

Using an initial estimate of $v = 0$ as starting value, (10) is iterated over the variables $v_i$ until $v$ has settled on the fixed point $v^*$. The solution is considered to have converged when the average distance between the MFA-variables in two consecutive iterations falls below a threshold value $\delta_v$,

$$\frac{1}{L} \sum_{i=1}^{L} |v_i^{\text{new}} - v_i^{\text{old}}| < \delta_v.$$

(14)

A value of $\delta_v = 0.01$, established by trial-and-error, has been used throughout this paper.

C. Pruning the Problem

By monitoring the conditional entropy estimate $H(s_i|\bar{v}_i)$, which comes almost for free, since the quantities needed to compute it are already available from the computation of (10), it is possible to prune the problem as the optimization progresses. When $H(s_i|\bar{v}_i) = 0$ no more information on $s_i$ can be gained from $\bar{s}_i$ and it is no longer necessary to calculate the $i^{\text{th}}$ value from (10). In practice, when $H(s_i|\bar{v}_i)$ falls below a threshold value $\varepsilon_{\text{prune}} > 0$ the variable $v_i$ is rounded to its nearest discrete value.

The reduction in cost function evaluations from pruning the problem is in the range 30% to 50% which is a decent speedup, considering the simplicity of the action.

D. Entropy-directed Deterministic Annealing

The EDDA algorithm discussed above is described in pseudo-code in Listing 1.

It should be pointed out that since the cost function at hand is a quadratic form $f(v) = v^T R v - 2p^T v + g$, a cost update strategy, previously presented in [14], makes it unnecessary to compute the full extent of the expression and the computational complexity of the algorithm can be reduced by an order of magnitude. The two approaches are still mathematically equivalent.

IV. ALGORITHM PERFORMANCE

All algorithms have been coded in C++ and rely on the free numerical analysis software GNU Octave[15]. They were run on a 1GHz Powerbook G4 with 1GB of RAM.
Algorithm 1 Entropy-directed Deterministic Annealing

\begin{verbatim}
Initialize state variables, \( v_i := 0, \ i = 1 \ldots L \)
Set initial control parameter value, \( c \)
Active set := 1 \ldots L
repeat
    \( v^{old} := v \)
    for \( i \) in active set do
        Calculate \( v_i := \sum s_i e^{-f(s_i, v_i)/c} / \sum s_i e^{-f(s_i, v_i)/c} \)
    end for
    until \( \frac{1}{L} \sum_i |v_i - v^{old}_i| < \delta_v \)
if \( H(s_i|v_i) < \varepsilon_{\text{prune}} \) then
    Remove \( i \) from active set
    \( v_i := \text{round}(v_i) \)
end if
\( c := \lambda c \)
until \( H(v) < \varepsilon_{\text{stop}} \)
\end{verbatim}

A. Example 1

To evaluate the performance of the EDDA algorithm, a design from [1], [2], [11] was chosen and the iterative Lagrange multiplier (ILM) algorithm from [1] and a SA algorithm has been implemented and are compared to EDDA.

The specification is a linear phase FIR filter with \( B = 10 \), passband \([0, 0.10625]\), stopband \([0.14375, 0.5]\) and weighting \( w(\omega) = \begin{cases} 1, & 0 \leq \omega < 0.103 \\ \sqrt{3}, & 0.103 \leq \omega \leq 0.10625 \\ \sqrt{8}, & 0.14375 \leq \omega < 0.15 \\ 1, & 0.15 \leq \omega \leq 0.5 \end{cases} \) (15)

The number of independent variables is \( M = (N + 1)/2 \). All NZT figures are given for the \( M \) independent coefficients, after a simple conversion to canonical form.

In the implementation of the EDDA algorithm, the parameters and thresholds were \( c_0 = 10^4 \), \( \lambda = 0.85 \), \( \delta_v = 0.01 \), \( \varepsilon_{\text{prune}} = 0.05 \), and \( \varepsilon_{\text{stop}} = \log_2 3 \).

In the ILM implementation, four quantized coefficients in the vicinity of a non-quantized coefficient were examined. The quantized coefficients were made up of at most two SPT terms.

The SA algorithm was run to estimate the best possible quantization and used (9) with \( \alpha = 0 \) as the target function. It used the same annealing schedule as EDDA, evaluating \( 100 \times N \) random state perturbations at every annealing step, stopping when < 3\% of the proposed changes were accepted.

The performance analysis of the EDDA algorithm is split over the following three sections.

1) \textit{Integrated Squared Error}: The filter design error \( f_0 \) for filters of odd \( N = 31, \ldots, 71 \) for each algorithm, are shown in Fig. 1. For the EDDA algorithm results for \( \alpha = 0 \) and \( \alpha = 50 \) are shown and, for reference, \( f_{\text{LS}} \) corresponding to the optimal continuous solution.

The results obtained by EDDA, using \( \alpha = 0 \), are of the same quality as those obtained by SA. The EDDA algorithm with \( \alpha = 50 \) performs better, in terms of filter design error, than the ILM algorithm except for filter lengths \( N < 45 \).

2) \textit{Non-zero SPT Term Reduction}: The ILM algorithm is different from EDDA and SA in that the former constrains each coefficient to have at most two SPT terms whereas the latter two imposes a soft constraint on the total number of non-zero SPT terms through parameter \( \alpha \).

In Fig. 2 the number of non-zero SPT terms in the solutions are shown and the impact of increasing \( \alpha \) from 0 to 50 in the EDDA algorithm is clearly visible as a substantial reduction of the number of non-zero SPT terms.

The quality of the EDDA solution, using \( \alpha = 50 \), in terms of \( f_0 \) is better than the ILM algorithm when \( N > 43 \). This is in line with the observation [3], [4], [16] that constraining the total number of SPT terms rather than the number of terms per coefficient results in better filter performance.

3) \textit{Solution Time}: Finally, the time taken to solve a problem is important since, if it is short enough, the designer can experiment with the design parameters and \textit{interactively} explore the design space. In Fig. 3 it can be seen that EDDA is slightly faster than ILM, both of which are more than an order of magnitude faster than simulated annealing.

![Fig. 1](image1.png)

![Fig. 2](image2.png)
Most notable is that the EDDA algorithm provides solutions of the same quality, in terms of $f_0$ and NZT, as those obtained by simulated annealing, but in a fraction of the time.

![Graph showing solution time versus filter length N for different algorithms](image)

**Fig. 3.** The solution time versus filter length $N$ for the different algorithms.

### B. Example 2

A halfband filter was specified to find out whether or not the EDDA algorithm would exploit that every other coefficient is zero. In reality, known coefficient values should be fixed and their corresponding variables taken out of the active set.

The specification was linear phase FIR, $B = 8$, passband $[0, 0.2]$, stopband $[0.3, 0.5]$ and $w(\omega) = 1$, $\omega \in [0, 0.5]$.

The optimization used $\alpha = 10$, $\lambda = 0.8$ and took 0.15 seconds. The result is shown in Fig. 4. The coefficients

$[128, 81, 0, -24, 0, 12, 0, -6, 0, 3, 0, -1, 0] \times 2^{-B}$

had a total of 13 non-zero SPT terms, but the realization can be reduced even further by exploiting that the sequence $[-24, 12, -6, 3]$ can be replaced by shifts and negations.

![Graph showing the halfband filter obtained from the EDDA algorithm](image)

**Fig. 4.** The halfband filter obtained from the EDDA algorithm.

### V. Summary and Conclusions

In this paper we have presented the entropy-directed deterministic annealing (EDDA) optimization algorithm applicable to the design of digital filters with discrete coefficients, each implemented as a sum of a signed powers-of-two terms. The algorithm operates on the bit level of the design and hardware realization aspects can thus be co-optimized with frequency response aspects.

The combination of a cost updating scheme, eliminating the need to repeatedly compute the full extent of the cost function, and progressive pruning of the problem leads to a very fast algorithm. The time taken to solve large problems has been reduced to a level where interactive design of digital filters, e.g., minimizing hardware demands by loosening the magnitude specification or maximizing filter performance given a certain hardware budget, is possible.

### References


