MetaCores: Design and Optimization Techniques

Seapahn Meguerdichian[1], Farinaz Koushanfar[2], Advait Mogre[3], Dusan Petranovic[3], Miodrag Potkonjak[1]
[1] Computer Science Department, University of California, Los Angeles
[2] Department of EE and CS, University of California, Berkeley
[3] LSI Logic Corporation, Milpitas, California
{seapahn, farinaz, miodrag}@cs.ucla.edu
{dusan, advait}@lsil.com

ABSTRACT

Currently, hardware intellectual property (IP) is delivered at three levels of abstraction: hard, firm, and soft. In order to further enhance performance, efficiency, and flexibility of IP design, we have developed a new approach for designing hardware and software IP called MetaCores. The new design approach starts at the algorithm level and leverages on the algorithm's intrinsic optimization degrees of freedom. The approach has four main components: *(i)* problem formulation and identification of optimization degrees of freedom, *(ii)* objective functions and constraints, *(iii)* cost evaluation engine, and *(iv)* multiresolution design space search. From the algorithmic viewpoint, the main contribution is the introduction of multiresolution search in algorithm optimization and synthesis process. We have applied the approach to the development of Viterbi and IIR MetaCores. Experimental results demonstrate the effectiveness of the new approach.

1. INTRODUCTION

The rapidly growing gap between silicon capacity and design productivity has resulted in a pressing need for design reuse. Hardware building blocks, usually under the name of cores, have become increasingly popular as the most efficient way of reusing design intellectual property (IP). While there exist several potential classification schemes for integrated circuits (IC) IP, the classification of cores according to their level of implementation details is by far the most popular.

Hard cores are IPs completely implemented using a particular physical design library. Firm cores are also completely implemented, including physical design, but are targeted at a symbolic library. Finally, soft cores are described in high level languages such as VHDL or Verilog. Clearly, while hard cores provide complete information about all relevant design parameters and facilitate the highest level of performance and implementation parameter optimization for the selected library, soft cores are superior in terms of their flexibility and application range. Initially, hard cores dominated the IP reuse market and practice, but recently there is an increasing trend toward other types of cores and in particular, soft cores. Additionally, parameterized, configurable, and programmable cores (such as Tensilica and Improv) have been rapidly gaining popularity.

We present a new approach to IC IP development. The approach can be viewed as a natural next step in core evolution because we consider design optimization and its suitability for efficient implementation at an even higher level than the high-level language specification. We analyze the algorithm for a particular application that is the target for creating the core with respect to its performance, and estimate implementation area and speed. Specifically, we identify the degrees of freedom for the algorithm alternations under specific targeted implementation objective functions and constraints. By searching the algorithm solution space, we identify the algorithm structure that is best suited for the specified design goals and constraints.

DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA

1.1. Trade-offs: Simple Example

Let us consider an example to introduce the key ideas discussed in this paper. Altering several key parameters in the Viterbi decoding algorithm used in convolutional forward-error-correction, can have tremendous impacts on the attributes of the final design. Although an experienced designer may successfully guess the general outcome of changing each parameter, initially, it is not always clear exactly what configuration is best suited for a specific application. As an example, Table 1 presents three different instances of the Viterbi decoder. Each instance is obtained by altering only a subset of the parameters that effect the Viterbi algorithm. The three main metrics that we use to evaluate the performance of the Viterbi decoder are bit-error-rate, throughput, and chip area. Table 1 lists the specific parameters used and the estimated area requirements for each instance when the desired throughput is fixed at 1 Mbps. Figure 1 shows the bit-error-rate (BER) curves obtained by software simulation for each case, under varying signal-to-noise ratios.

Although all three cases exhibit comparable BER curves as shown in Figure 2, each can have drastically different area requirements when the desired throughput is fixed as shown in Table 1. Here we have selected only a few parameters while in general, the solution space is very large and complex.

K	Trellis Depth	Quantization Bits Low / high	Multi-res. Paths	Area mm^2
3	2	3 / NA	NA	0.26
5	5	1 / 3	8	0.56
7	5	1 / 3	4	1.73

Table 1. Area Estimates of Three Instances of the Viterbi Decoder Under Fixed Throughput (1Mbps)

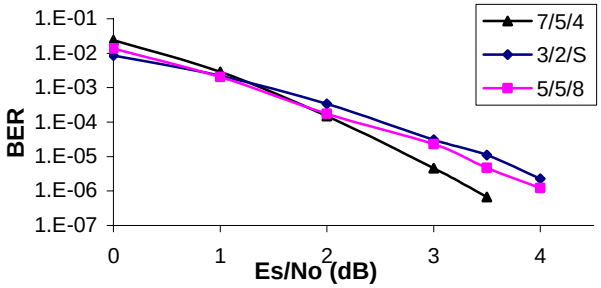

Figure 1. Viterbi Decoder BER vs. Signal-to-Noise Ratio

1.2. Goals

Our primary objective is to demonstrate the importance and effectiveness of leveraging the potential of algorithm design through performance simulation and area and speed estimation. Note that performance indicates quantified qualities from the application point of view. For example, in error correction, algorithm performance is measured by the bit error rate while in lossy compression, the level of compression and loss rate are used to measure performance.

We also introduce and demonstrate the first effective quantitative algorithm design technique. Also, the study of algorithmic degrees of freedom of the Viterbi decoder has an interesting and potentially important side result: a new multiresolution Viterbi decoding algorithm. Our final objective and result is to produce a high performance, low cost, Viterbi decoder implementation for a variety of bit error rates (BER) and throughput requirements. While the Viterbi is the primary driver in our discussions, we also demonstrate the methodology when applied to an IIR filter as a validation example.

1.3. Paper Organization

The reminder of the paper is organized as follows. We first summarize the related work. Then, we survey all preliminary material about the Viterbi algorithm and IIR filters, the targets of our case studies, and discuss area and timing models. Section 4 is the backbone of the paper and presents detailed description of the problem formulation, objectives, and the new multi-resolution design space search technique. Finally, we demonstrate the effectiveness of the meta-core design approach using the Viterbi decoder driver example.

2. RELATED WORK

Although, IC component reuse has been widely practiced at many design centers since the beginnings of silicon designs, in the last few years a strong consensus has formed that IP reuse will be a dominant enabling force for the future generation of designs. A number of design companies have been making strong efforts to develop their IP portfolio, often mainly for internal use. There are also several companies (e.g. Artisan) who have completely based their business model on providing design IP. Thus, IP creation, assembly, and testing have received significant recent research attention [Sch99, Del99, Zor99].

The Viterbi decoding algorithm has by far been the most widely studied and used convolutional error correction code in both wired and wireless communication. It is considered a fundamental DSP algorithm and the performances of modern DSP chips are often quoted in terms of their Viterbi decoding speed. The algorithm is well described in the classical papers. Several hard and soft Viterbi decoder cores have been reported in [Smi98, Bur99].

Filtering (processing) on streams of data is a fundamental task widely used in digital signal processing, communication, and control applications [Opp89]. Infinite Impulse Response (IIR) filters are particularly attractive due to their relatively low implementation complexity. A variety of different topological structures have been proposed for the realization of IIR filters, including direct form, cascade, parallel, continued fraction, ladder, Wave digital, state-space digital, orthogonal and multi variable digital lattice. Today, in addition to many public domain IIR filter design programs, there are also well-supported commercial design tools for synthesis of IIR filters including MATLAB and SPW.

Algorithm selection and design has been a popular research topic in a number of research fields, particularly in artificial intelligence where four main directions have emerged: first order logic-based methods [Gre69], rewrite systems [Der85], transformational approaches [Dar81] and schema-based programming [Wil83]. While the proposed techniques are strategically and algorithmically very different, they all share a common weakness in their inability to scale to problems of practical importance. Several VLSI DSP efforts have also addressed the algorithm selection and design process [Opp92, Bro97, Pro96]. In a sense, the algorithm design aspect of our approach is most similar to one presented in [Pot99]. However, while their approaches are limited just to the algorithm selection process, we explore the much richer design space where a number of vital algorithm parameters are considered. Furthermore, a major distinction of this approach with respect to all surveyed work is that our main goal is the development of IP creation techniques at the higher level of abstraction.

Multiresolution techniques have been popular for a long time, in particular in image, video, and in general, digital signal processing. The popularity of multiresolution techniques in DSP has been further amplified with the introduction of wavelet transforms [Mal98]. Multiresolution techniques have also been widely used in numerical algorithms and in mesh-based finite element techniques.

3. PRELIMINARIES

3.1. Viterbi Algorithm

In most modern communication systems, channel coding is used to increase bandwidth, add error detection and correction capabilities, and provide a systematic way to translate logical bits of information to analog channel symbols used in transmission. Convolutional coding and block coding are the two major forms of channel coding used today. As their names imply, in convolutional coding the algorithms work on a few bits at a time while in block coding big chunks of data are processed together. Generally, convolutional coding is better suited for processing continuous data streams with relatively small latencies. Also, since convolutional forward error correction (FEC) works well with data streams affected by the atmospheric and environmental noise (Additive White Gaussian Noise) encountered in satellite and cable communications, they have found widespread use in many advanced communication systems. Viterbi decoding is one of the most popular FEC technique used today and is therefore the main focus of our discussion.

Convolutional codes are usually defined using the two parameters, code rate (k/n) and constraint length (K). The code rate of the convolutional encoder is calculated as the ratio k/n where k is the number of input data bits and n is the number of channel symbols output by the encoder. The constraint length K is directly related to the number of registers in the encoder. These (shift) registers hold the previous data values that are systematically convolved with the incoming data bits. This redundancy of information in the final transmission stream is the key factor enabling the error correction capabilities that are necessary when dealing with transmission errors. Figure 2 shows an example of a ½ rate encoder with $K=3$ (4 states).

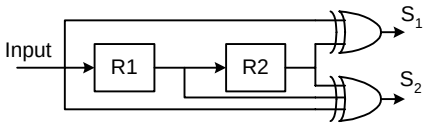

Figure 2. Convolutional Encoder

The simple encoder in Figure 2 generates two channel symbols as each incoming data bit is shifted into register R_1. The connections from the registers to the output XOR gates are defined by the polynomial G. There are many studies that show the optimal K and G in different situations [Lar73, Ode70]. It is interesting to note that although the ½ rate encoding effectively reduces the

channel bandwidth by a factor of two, the power savings that are gained due to the increased reliability of the channel offset the negative effects of the reduced bandwidth and overall, the technique improves the efficiency of the channel.

Viterbi decoding and sequential decoding are the two main types of algorithms used with convolutional codes. Although sequential decoding performs very well with long-constraint based convolutional codes, it has a variable decoding time and is less suited for hardware implementations. On the other hand, the Viterbi decoding algorithm developed by Andrew J. Viterbi, one of the founders of Qualcomm Corporation [Vit67], has fixed decoding times and is well suited for hardware implementations. The exponentially increasing computation requirements as a function of constraint length (*K*) limit current implementations of the Viterbi decoder to about *K=9*.

3.2. Viterbi Decoder

Viterbi decoding, also known as maximum-likelihood decoding, is comprised of the two main tasks of updating the trellis and trace-back. The trellis used in Viterbi decoding is essentially the convolutional encoder state transition diagram with an extra time dimension. Figure 3 shows an example of a trellis diagram for the *4*-state (*K=3*) Viterbi decoder. The four possible convolutional encoder states are depicted as four rows in the trellis. The solid edges represent transitions based on *1* inputs and the dashed lines represent transitions based on *0* inputs. There are two channel symbols produced by the encoder associated with each branch in the trellis.

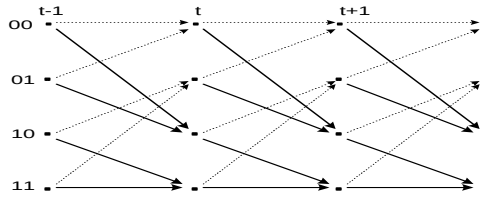

Figure 3. Viterbi Trellis Diagram

After each time instance *t*, the elements in the column *t* contain the accumulated error metric for each encoder state, up-to and including time *t*. Every time a pair of channel symbols is received, the algorithm updates the trellis by computing the branch metric associated with each transition. In hard decision decoding, the branch metric is most often defined to be the Hamming distance between the channel symbols and the symbols associated with each branch. So for hard decision ½ rate decoding (*2* channel symbols per branch), the possible branch metric values are *0*, *1*, and *2*, depending on the number of mismatched symbols. The total error associated with taking each branch is the sum of the branch metric and the accumulated error value of the state from which the branch initiates. Since there are two possible transitions (branches) into each state, the smaller of the two accumulated error metrics is used to replace the current value of each state.

The state with the lowest accumulated error metric is chosen as the candidate for trace-back. The path created by taking each branch leading to the candidate state is traced back for a predefined number of steps. The initial branch in the trace-back path indicates the most likely transition in the convolutional encoder and can therefore be used to obtain the actual encoded bit value in the original data stream.

To make the decoder work, received channel symbols must be quantized. In hard decision decoding, channel symbols can be either *0* or *1*. Hard decision Viterbi decoders can be extremely fast due to the small number of bits that are involved in the computations. However, tremendous BER improvements have been achieved by increasing the number of bits (resolution) used in quantizing the channel symbols. Figure 4 shows an example of a uniform quantizer using 3-bits (8 levels) to represent a symbol received on the channel [Aha95]. The ratio E_s/N_0, the energy per symbol to noise density ratio, is used to calculate *D*, the decision level. For a detailed discussion of quantization methods and their effects on the Viterbi decoding algorithm refer to [Aha95].

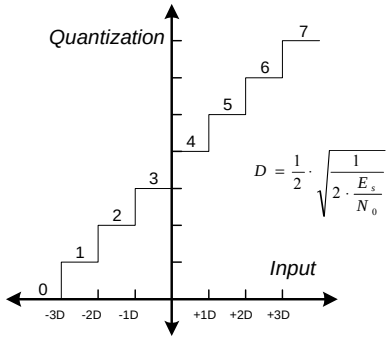

Figure 4. Adaptive Soft Quantization

3.3. Multiresolution Viterbi

The benefits of soft decision over hard decision decoding are offset by the cost of significantly bigger and slower hardware. Here we propose a new multi-resolution Viterbi decoding algorithm that exhibits the BER performance benefits of higher resolution soft decoding while maintaining minimal area and delay overheads. The multi-resolution Viterbi decoding method is based on the key observation that at any given time, only a relatively small number of the trellis states are possible candidates for trace-back while others with larger accumulated errors are less likely to be useful. We use this observation and update the trellis using fewer bits and after each step recalculate the branch metrics for several of the "better" paths (paths with smaller accumulated errors) using higher precision.

Since in trace-back the state with the minimum accumulated error is chosen as the starting point, the algorithm must be designed such that no state is given an unfair advantage over the others. The higher precision recalculation of branch metrics for the most likely candidate states improves the probability of selecting the real best state for trace-back. However, since the quantization and branch error calculation methods are different for each case, a correction term must be added to the recalculated branch metrics to keep the accumulated error values normalized.

There are several methods of normalizing the lower and higher resolution branch metric values obtained during decoding. In general, an efficient approach of finding the correction value is by calculating the difference between the best high resolution and the best low resolution branch metric at each iteration. We can further improve on this approach by averaging the differences of two or more branch metrics. Experimental results show that big improvements in performance can be achieved over hard decision decoding by only recalculating a small fraction of the trellis paths.

3.4. Infinite Impulse Response (IIR) Filters

The functionality of IIR filter can be compactly and completely captured by its transfer function. Figure 5 shows a typical transfer function for a low-pass IIR filter. There are several parameters that characterize a filter that include passband and stopband frequencies, passband ripple, stopband attenuation, *3*-dB bandwidth, and gain. Note that all of the structures mentioned in Section 2, can be used to implement an arbitrary transfer function. However they greatly differ in terms of hardware requirements, such as number of multiplications, number of additions, word

length, interconnect, and registers. It has been shown that depending on the required throughput, various structures are the better options for implementation [Pot99].

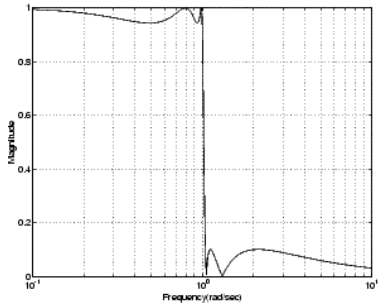

Figure 5. Typical transfer function for low pass IIR filter (Elliptic IIR Filter)

4. DESIGN FLOW

4.1. Problem Formulation and Optimization Degrees of Freedom – Viterbi Decoder

There are many parameters that can effect the performance of the Viterbi decoder. Currently, we model the domain of the solution space as an 8-dimensional matrix. Table 2 lists all parameters that constitute the degrees of freedom in our solution space.

K	Constraint length {3,4,5,6,7,…}
L	Trace-back depth {1*K, 2*K, 3*K, 4*K, 5*K, …}
G	Encoder Polynomial(s)
R_1	Quantization used for low-resolution decoding
R_2	Quantization used for high-resolution decoding (multi)
Q	Quantization method (hard, fixed, adaptive)
N	Normalization method
M	Number of multi-resolution paths $(1,2,\ldots,2^{K-1})$

Table 2. Viterbi Decoder Parameters

The parameter *K* is the constraint length of the convolutional encoder and *L* is the trace-back depth of the decoder. Although *K* and *L* do not have any theoretic bounds, we limit our search to current practical values of $K<10$ and $L<30*K$. Our experiments have shown that in most cases, trellis depths larger than $7*K$ do not have any significant impact on BER. There are several standard specifications of the encoder polynomial *G* for different values of *K*. The user has the option of selecting multiple variations of *G* to be included in the search, although in most cases *G* is fixed. The quantization resolution parameters R_1 and R_2 indicate the number of bits used in the calculation of the trellis branch metrics. As we discussed earlier, higher number of bits (soft decision) translate to better BER performance. Also, the choice of the quantization resolution parameters R_1 and R_2, effect the multiresolution normalization method *N*. Currently, *N* specifies the number of branch metric values used in the calculation of the multi-resolution correction factor. For pure hard or soft decoding this parameter is set to 0 and for multiresolution decoding $1 \le N \le M$. The parameter *M* specifies the number of trellis states (paths) that are recalculated using higher resolution in multi-resolution decoding.

4.2. Objective Function and Constraints

The performance of each instance of the Viterbi decoder is quantified in terms of the following three metrics: *(i) Bit Error Rate (BER) (ii) Area,* and *(iii) Throughput.* Software simulation is used to measure the BER of each instance of the algorithm under varying signal to noise ratios. Generally, the user defines a threshold curve that serves as a guide for the desired BER performance. Area and throughput metrics are obtained by simulating the algorithm using *Trimaran* [Tri99]. Trimaran provides a compiler and hardware platform for parallel programmable VLIW and Superscalar architectures. We use Trimaran to estimate the area requirements of each candidate solution for a fixed throughput. To evaluate each instance, we generate the source code that Trimaran can compile and optimize and specify the Trimaran hardware architecture parameters such as register file sizes, memory hierarchy, number of arithmetic logic units (ALU) and others. During the simulation, Trimaran collects several statistics for each solution instance including the total number of operations executed (load, store, ALU, branch, etc.) the total number of cycles required to complete the decoding task for a fixed number of bits, dynamic register allocation overhead, and several others. Using our Trimaran area models, we then obtain the area requirements of each instance based on the desired throughput (clock rate).

4.3. Hardware Area and Throughput Model

We use the LSI Logic TR4101 microprocessor as the basis for our model for Trimaran hardware due to the similarities between the two architectures. This processor has a feature size of $0.35\mu m$ running at a maximum clock speed of 81MHz. We use the quadratic scaling factor

$$\lambda = \left(\frac{\alpha}{0.35}\right)^2 \cdot data\_path\_factor$$

to scale the area to an architecture based on a feature size of $\alpha\,\mu m$. Since the TR4101 is a 32-bit processor, we use the *data_path_factor* obtained from [Erc98] to adjusts the area requirement based on the width of the data path (number of bits).

In our area model, we assume that clock rates scale linearly with feature size with smaller sizes resulting in faster clock rates. Also, to account for different data-path sizes, we use scaling factors based on data presented in [Erc98] to adjust the clock rate.

4.4. Multiresolution Search

There are roughly 10^8 distinct points in the solution space we have defined for the Viterbi Decoder and even more options for the IIR filter. Due to the large size of the solution space, exhaustive search methods are ineffective. We use a multiresolution search technique to search the solution space in an efficient manner by concentrating our efforts on promising regions. We initiate the search on a fixed grid in the solution space. For example, in the case of the Viterbi decoder, since we have defined 8 dimensions, we decide to evaluate up to 256 instances. However, in most practical cases this number is much lower since some of the parameters are fixed (e.g. *G*, *N*). Using the performance evaluated at each point on the grid as a guide, we further search the regions that are most promising in terms of area, throughput, and BER using a finer grid and more accurate simulation results (longer run times).

```
R = Initial search resolution
G = Initial sparse search points
Procedure Viterbi_Metacore_Search(G,R) {
   For each p_i ∈ G
      p_i.BER = Simulate and measure BER
      Find p_i.Area using Trimaran using given p_i.Throughput
   End For
   NewGridSet = Refine_Grid(G)
   NewR = R+Resolution_Increment
   If (NewR<Max_Search_Resolution)
      For each G_i∈NewGridSet
         Viterbi_Metacore_Search(G_i,NewR)
}
```

Figure 6. Pseudo Code for Viterbi Metacore Design

The pseudo code in Figure 6 describes the Viterbi Metacore search algorithm. When calculating the new grid (*Refine_Grid*) regions, we extract regions enclosed by the points that are more likely to contain promising solutions. Since our area and throughput functions are smooth and continuous, we use interpolation between the points on the grid to calculate initial estimates. However, BER is probabilistic by nature and interpolation can lead to inaccurate conclusions especially if simulation times are kept short. We use Bayesian probabilistic techniques to assign a BER probability to each point $p_i \in G$, based on the BER values of its neighbors. Essentially, we use conditional probabilities about observed dependencies in the solution space points to predict most likely value at points that are still to be considered during the search. The search is then recursively executed on the newly formed regions with higher resolution to find and refine the best candidate solutions.

In general, we classify the design space parameters as: *(i)* discrete or continuous and *(ii)* correlated or non-correlated. We further distinguish the correlated parameters using their structures such as monotonic, linear, quadratic, probabilistic, etc. Clearly, non-correlated parameters are more difficult to handle since optimal solutions cannot be found as rapidly using our heuristic techniques. Also, the search method presented above for the Viterbi Metacore design is clearly greedy. This design choice is justified by the speed of the searching mechanism and ease of implementation. However, the optimality of the search and the results can be increased using longer simulation times and relaxing the search space pruning technique at the cost of significantly longer runtimes.

4.5. Validation Example – IIR Design flow

There are several parameters that impact the performance and the computational complexity of IIR filters. Here, we consider the following degrees of freedom: topological structure, number of stages, word length, and passband ripple characteristics. The performance of an instance of an IIR filter is measure using the following criteria: *(i)* 3-dB bandwidth, *(ii)* area, *(iii)* throughput, and *(iv)* latency. SPW software simulations are used to measure gain, 3-dB bandwidth, pass band ripple, and stop band attenuation characteristics. Area, throughput, and latency are obtained using the HYPER behavioral synthesis tools [Rab91]. Specifically we use HYPER tools for early estimation of both active logic area (execution units, registers, and interconnect) as well as statistical tools for prediction of total area. The final implementation is obtained using Hyper and Lager tools [Rab91].

To evaluate each candidate for implementation, we start by entering user specified transfer functions in SPW and consequently generate Silage code which is used as input to the HYPER behavioral synthesis tool. HYPER also outputs timing information such as the length of the clock cycle and the number of cycles used. This information is used to compute throughput and latency.

5. EXPERIMENTAL RESULTS

5.1. Experimentation Platform – Viterbi Metacore

The main user interface, multiresolution search algorithm, and the multiresolution Viterbi decoder simulator are implemented as a Microsoft Windows™ application using Visual C++ 6.0 IDE. The Trimaran environment is set up on an Intel Pentium III based PC running RedHat Linux 6.1. This configuration facilitates the parallel execution of the Viterbi software and hardware simulations.

Figure 7 shows a screen capture of the main user interface in action, where the user can specify most of the algorithmic and hardware related parameters. Several configuration files and scripts

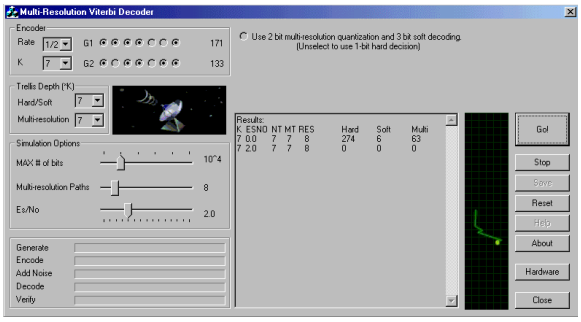

Figure 7. Screenshot of Main User Interface

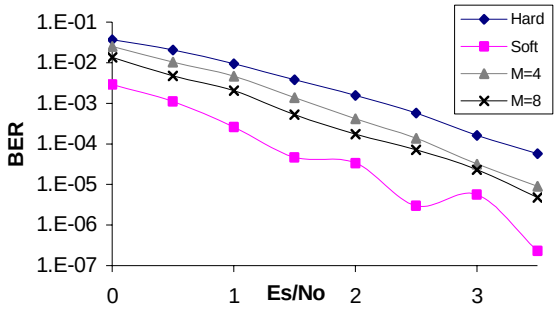

Figure 8. BER vs. Signal-to-Noise Ratio for Viterbi Decoder (*K=5, L=5 , R_1=1, R_3=3 with adaptive quantization)*

are used to specify the range of parameters used and automate user tasks.

5.2. Experimental Results – Viterbi Decoder

We present several experimental results to better illustrate the general trade-offs involved in our approach, specifically in the exploration of the multi-resolution Viterbi decoder Metacore. The graph in Figure 8 shows the relative BER for hard, soft, and multiresolution Viterbi decoding with *K=5*, using *1*-bit low-resolution and *3*-bit adaptively quantized high-resolution decoding. In this case, on average, using *4* high-resolution paths (*M=4*) results in a *64%* improvement in BER while using *8* high-resolution paths (*M=8*) results in *82%* improvement over pure hard-decision decoding.

Table 3 lists the results of several Metacore search outcomes using different parameter specifications. In each case, the BER and throughput were specified. Normalization (*N*) and polynomial (*G*) were fixed to speedup the search process. The estimated area requirement and the associated Viterbi metacore parameters are reported for comparison.

5.3. Experimental Results – *IIR Filter*

In order to evaluate the effectiveness of the proposed Metacore design technique on the IIR designs, we specified a bandpass IIR filter with the following characteristics: $\omega_{p1}=0.411111\pi$, $\omega_{p2}=0.466667\pi$, $\varepsilon_p=0.015782$, $\omega_{s1}=0.3487015\pi$, $\omega_{s2}=0.0.494444\pi$, $\varepsilon_s=0.0157816$, where ω_{p1} and ω_{p2} are the bandpass frequencies, ω_{s1} and ω_{s2} are stopband frequencies, ε_p is the passband ripple, and ε_s is the stopband ripple (assuming a standard normalized filter characteristics).

Table 4 shows the experimental results after applying the multiresolution search algorithm on the IIR filter. The first column shows the throughput of the filter (μs). The second column indicates the best solution in terms of the area. The third column shows the average case solution (in terms of area). Column four, indicates

Desired BER *(at E_s/N_0=1.0)*	Desired Throughput	K	L *(*K)*	G	R_1 *(bits)*	R_2 *(bits)*	Q*	N	M	Area *(mm^2)*
$1x10^{-2}$	5 Mbps	3	4	7,5	2	NA	A	1	NA	0.35
$1 x10^{-4}$	2 Mbps	5	6	35,23	1	3	F	1	5	1.2
$1x10^{-5}$	1 Mbps	7	7	171,133	3	NA	A	1	NA	2.2
$1x10^{-5}$	3 Mbps	7	7	171,133	2	4	A	1	NA	3.3
$1x10^{-9}$	1 Mbps	x	x	x	x	x	x	x	x	Not Feasible

Table 3. Viterbi Decoder results with several BER and Throughput Requirements (*A=Adaptive, F=Fixed)

the percentage improvement in the area after applying the Metacore optimization in comparison with the average case. The last column of the table shows the structure of the filter that was used to produce the best implementation. The average and median reduction in area over all designs generated during the search process were 75.12% and 71.92% respectively.

6. CONCLUSION

We presented a new approach for designing hardware and software IP that starts at the algorithm level and leverages on algorithm's intrinsic optimization degrees of freedom. The main components of the approach namely problem formulation and identification of optimization degrees of freedom, objective functions and constraints, cost evaluation engine, and multiresolution design space search were discussed in detail with illustrative examples from our chosen application, the Viterbi decoder and our validation example, the IIR filter.

We demonstrated the multiresolution design space search for optimization and synthesis and presented parameter classifications and the tradeoffs involved. Furthermore, we presented the new multi-resolution Viterbi decoding algorithm and illustrated its capabilities as a viable alternative to pure hard and soft decision decoders.

Throughput *(μs)*	Multi-res. Area *(mm^2)*	Average Area *(mm^2)*	Reduction *%*	Structure
5	5.73	15.75	63.62	Ladder
4	5.92	18.27	67.60	Parallel
3	5.92	19.94	70.31	Parallel
2	5.92	21.08	71.92	Parallel
1	6.11	35.81	82.94	Cascade
0.5	11.63	69.98	83.39	Cascade
0.25	22.14	158.90	86.07	Cascade

Table 4. Performance of IIR Filter

REFERENCES

[Aha95] Advanced Hardware Architectures Inc., "Soft Decision Thresholds and Effects on Viterbi Performance," ANRS07-0795, http://www.aha.com, 1995.

[Bro97] D.W. Brown, J.G. McWhirter, "The Design Of A Block-Regularized Parameter Estimator By Algorithmic Engineering," *International Journal of Adaptive Control and Signal Processing*, Vol. 11, No. 5, pp. 381-393, 1997.

[Bur99] R. Burger, G. Cesana, M. Paolini, M. Turolla, "A Fully Synthesizable Parameterized Viterbi Decoder," *Custom Integrated Circuits Con*ference, pp. 27-30, 1999.

[Dar81] J. Darlington, "An Experimental Program Transformation And Synthesis System," *Artificial Intelligence*, Vol. 16, pp.1-46, 1981.

[Del99] M. Delpasso, A. Bogliolo, L. Benini, "Virtual Simulation Of Distributed IP-Based Designs," *36-th DAC*, pp. 50-55, 1999.

[Der85] N. Dershowitz, "Synthesis By Completion," *The Ninth Int. Joint Conference on Artificial Intelligence*, pp. 208-214, 1985.

[Erc98] M. Ercegovac, D. Kirovski, M. Potkonjak, "Behavioral Synthesis Optimization Using Multiple Precision Arithmetic," *ICASSP '98*, p.3113-16, 1998.

[Gre69] C. Green, "Application Of Theorem Proving To Problem Solving", *Proc. Of the First International Joint Conference on Artificial Intelligence*, pp. 219-239, 1969.

[Lar73] K. J. Larsen, "Short Convolutional Codes With Maximal Free Distance For Rates 1/2, 1/3, And 1/4," *IEEE Trans. on Inf. Theory*, vol. IT-19, pp. 371-372, 1973

[Mal98] S. G. Mallat, *A Wavelet Tour Of Signal Processing*. San Diego : Academic Press, 1998.

[Ode70] J. P. Odenwalder, "Optimum Decoding Of Convolutional Codes," Ph.D. Dissertation, Department of Systems Sciences, UCLA, 1970.

[Opp89] A.V. Oppenheim, R.W. Shafer, *Discrete-Time Signal Processing,* Prentice Hall, Englewood Cliffs, NJ, 1989.

[Opp92] A.V. Oppenheim, S.H. Nawab, *Symbolic and Knowledge-based Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[Pot99] M. Potkonjak, J.M. Rabaey, "Algorithm Selection: A Quantitative Optimization-Intensive Approach," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.18, (no.5), IEEE, May 1999.

[Pro96] I.K. Proudler, J.G. McWhirter, M. Moonen, G. Hekstra, "Formal Derivation Of A Systolic Array For Recursive Least Squares Estimation," *Trans. On Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 43, No. 3, pp.247-254, 1996.

[Rab91] J.M. Rabaey, C. Chu, P. Hoang, M. Potkonjak, "Fast Prototyping Of Datapath-Intensive Architectures," *IEEE Design & Test of Computers*, vol.8, June 1991. p.40-51.

[Sch99] P. Schaumont, R. Cmar, S. Vernalde, M. Engels, and others. "Hardware Reuse At The Behavioral Level," *DAC*, pp. 784-789, 1999.

[Smi98] B. Smith, J.V. McCanny, "Rapid Design Of High Performance Adaptive Equalizer And Viterbi Decoder For The Class-IV PRML Channel," *Workshop on Signal Processing Systems,* SIPS 98, pp. 307-316, 1998.

[Stu91] B. Stuckman, P. Scannell, "A Multidimensional Bayesian Global Search Method Which Incorporates Knowledge of an Upper Bound", Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics. pp. 591-596, 1991.

[Tri99] Trimaran: An Infrastructure for Research in Instruction-Level Parallelism, www.trimaran.org.

[Vit67] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. on Inf. Theory, vol. IT-13, pp.260-269, 1967

[Wil83] D.S. Wile, "Program Developments: Formal Explanations of Implementations", Comm. of the ACM, Vol. 26, No. 11, pp. 902-911, 1983.

[Zor99] Y. Zorian, E.J. Marinissen, S. Dey, "Testing embedded-core-based system chips", Computer, vol.32, pp. 52-60, 1999