Discrete-Relaxation-based Heuristic Techniques for Video Algorithm/Architecture Matching and System Level Transformations

Miodrag Potkonjak
C&C Research Laboratories, NEC USA, Princeton, NJ

Abstract: System level design and computational transformations have been rapidly establishing themselves as important high impact design steps which often have the most influential impact on the most important final performance parameter, throughput, of a design. In this paper, we introduce an iterative heuristic approach for throughput optimization when algorithm-architecture matching, and any of three behavioral transformations, retiming, rephasing, and pipelining, are considered simultaneously. The effectiveness of the approach and optimization algorithms is demonstrated on several video and image processing examples.

1.0 Motivation and Related Work

System level design and behavioral transformations have been rapidly establishing themselves as key design steps with the most influential impact on key final performance metrics, throughput and latency, of a design [Gup94, Wol94, Pot94a]. While current system level methods and CAD tools target a variety of application domains, such as hard-real time systems, embedded systems, and control applications, it is more and more apparent that DSP, and in particular image and video processing, applications are the most amenable domain for treatment using system level synthesis techniques and transformations [Wol94, Pot94a].

This is so because image and video processing are characterized by very high computational complexity, complex hierarchical structures, and simultaneously high I/O and memory requirements which makes their manual optimization difficult. The video and image processing tasks can be often described using synchronous data flow computational model [Lee87]. This model is exceptionally well suited for the application of optimization techniques due to its deterministic and well structured nature [Lee87, Lee95]. On many video and image applications are also often imposed intrinsic high throughput rates and the applications are often subject to low production cost and low power constraints due to the nature of modern consumer electronics market and portability requirements.

In this paper we study throughput optimization at the system level. The related work to the efforts presented in this paper can be traced along two lines of research: computational transformations and algorithm/architecture matching.

Transformations are changes in the structure of a computation so that the initially specified functional dependencies are preserved. Transformations are used for optimization of variety of different metrics in several computer science and engineering domains, including compilers, logic synthesis, and behavioral synthesis [Pot94].

The problem of throughput optimization using algorithm selection was recently introduced as an optimization challenge in [Pot94a] where improvements by a factor of 2 in throughput was reported on three smaller benchmark examples. Potkonjak and Rabaey also showed significantly higher effectiveness of algorithm selection for optimization of other design metrics, such as area, power, and fault-tolerance overhead. [Pot94a] The scope of this hardware-software codesign problem was significantly broadened recently in [DeS95]. They consider both algorithm and architecture selection, and more importantly, simultaneously considered also retiming. The optimal, worst case exponential run-time, ILP-based solution yielded significant improvements on five smaller examples.

In this paper, we consider also simultaneous algorithm/architecture matching. However, we enlarge the set of considered transformations and consider rephasing [Pot95]. We also recognize the limitations of the ILP-based solution and provide a fast and effective heuristic alternative. The larger and more powerful set of transformations enables higher level of optimization and the heuristic algorithms enables a significantly large application domain of the exploration of solution space at the system level.

2.0 Problem Formulation, Complexity, and Solution

Rapid progress in VLSI IC design methodologies and implementation technologies provides a system designer with numerous architectural platforms such as programmable video processors, DSP general purpose processors, microprocessors and custom ASIC. Furthermore, for each type of an implementation platform for a given application

0-8186-7310-9/95 $4.00

task there are numerous algorithmic options. For example, for image compression several methods are currently advocated, including transform coding [Rao90], predictive coding, vector quantization, model-based techniques [Aiz95], fractal-based techniques [Bar93], adaptive morphological subband decomposition [Egg95], neural networks-based approaches [Don95], simulated and mean filed annealing [Ozc95], and wavelets. Even, when application has only one

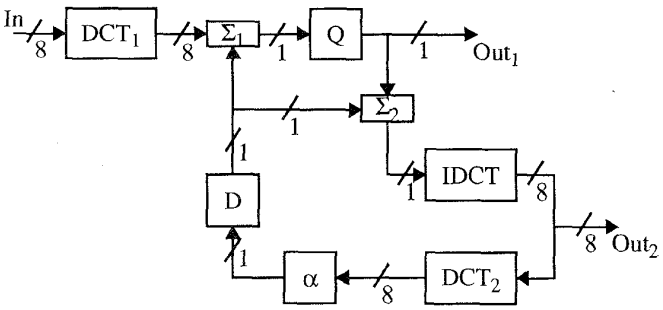

Figure 1. Throughput Optimization using Simultaneous Algorithm/Architecture Matchning and Rephasing

computational block, it has been shown that algorithm and architecture selection has very high impacts on the quality of the final implementation. For example, it has been shown that by selecting a proper fast DCT algorithm or a proper filter structure an order of magnitude improvement in several design metrics are achievable under the identical throughput and other implementation constraints [Pot94a]. This effect has even high impact when hierarchical designs are considered and when simultaneously the effects of retiming, rephasing, and pipelining are considered.

We introduce problem of throughput optimization by employing algorithm/architecture matching and rephasing and pipelining using the following example. Figure 1 shows the CDFG of the discrete cosine transform-based differential pulse coded modulation (DCT/DPCM) coding scheme with prediction in the transform domain [Jai81]. Σ denotes vector sum, DCT and IDCT direct and inverse discrete cosine transforms respectively, α is scalar - vector multiplication. The number next to the each edge denotes the number of inputs/outputs associated with the blocks to which the edge is associated. D denotes a hierarchical delay. Delays (states) are used to store data transferred between successive iterations and initial values.

If each input edge of a block has k hierarchical delays, the functionality of the design is not altered if those delays are deleted and replaced by k hierarchical delays on each output of the block or vice versa. This delay manipulation technique is called retiming [Lei83]. Pipelining is a behavioral transformation primarily used for throughput optimization. Pipelining adds the equal number of delays on each primary input or on each primary output. Pipelining is exceptionally powerful in designs which have no or only few small feedback parts. Recently a new transformation, rephasing, has been introduced [Pot95]. Rephasing is a timing transformations which assigns relative timing positioning of the states variables and the inputs in a computation. Rephasing provides all power of retiming [Lei83] and has several additional comparative advantages [Pot95], such as automatic elimination of granularity and I/O bottlenecks. Although the nature of rephasing and retiming is very different, many of retiming optimization algorithms can be easily modified when rephasing is used as one of CDFG optimization mechanisms.

Figure 1 illustrates the role of delays (states) in a hierarchical computations for breaking feedback loops and establishing the proper timing relationship among computational nodes. Note that the delay are used to break two feedback loops in the example:

$$\Sigma_1 \rightarrow Q \rightarrow \Sigma_2 \rightarrow IDCT \rightarrow DCT_2 \rightarrow \alpha$$

and

$$\Sigma_2 \rightarrow IDCT \rightarrow DCT_2 \rightarrow \alpha .$$

Each block in Figure 1 can be implemented using a variety of algorithms (CDFGs) and architectures. For example, for discrete cosine transform (DCT) options include the following fast algorithms: Lee's, Wang's, decimation in frequency (DIF), decimation in time (DIT), QR-planar rotation, Givens planar rotation, Arai's algorithm, and direct DCT [Rao90]. In addition, for each of the algorithms there are numerous implementation platforms, such as microprocessors, DSP processors, video processors, and custom ASIC. Equations (1) and (2) show the distances between each pair of nodes for two among several other DCT algorithm/architecture matching. The matrices are formed in a such way that element a_{ij} represents the distance between input i and output j. Note, that distances depend not only on the algorithm, but also on the selected architecture-implementation platform.

The goal is to select an algorithm/architecture pair for each building block of the hierarchical CDFG so that after the application of retiming, rephasing, or pipelining, the final maximum throughput (sampling rate) is maximized. It is important to note that there is a dependence between any of the considered transformations and algorithm/architecture selection. Therefore, the successive resolution of the two

$$DCT_i = \begin{bmatrix} 4 & 5 & 6 & 6 & 8 & 8 & 8 & 8 \\ 4 & 5 & 6 & 6 & 8 & 8 & 8 & 8 \\ 4 & 5 & 6 & 6 & 8 & 8 & 8 & 8 \\ 4 & 5 & 6 & 6 & 8 & 8 & 8 & 8 \\ 4 & 5 & 6 & 6 & 8 & 8 & 8 & 8 \\ 4 & 5 & 6 & 6 & 8 & 8 & 8 & 8 \\ 4 & 5 & 6 & 6 & 8 & 8 & 8 & 8 \\ 4 & 5 & 6 & 6 & 8 & 8 & 8 & 8 \end{bmatrix} \quad (1)\ DCT_{ii} = \begin{bmatrix} 4 & 5 & 5 & 4 & 5 & 8 & 5 & 8 \\ 4 & 5 & 5 & 4 & 6 & 8 & 5 & 5 \\ 4 & 5 & 6 & 4 & 4 & 8 & 5 & 5 \\ 4 & 5 & 7 & 4 & 4 & 8 & 5 & 5 \\ 4 & 5 & 5 & 4 & 6 & 8 & 5 & 5 \\ 4 & 5 & 5 & 4 & 6 & 8 & 5 & 5 \\ 4 & 5 & 6 & 4 & 7 & 8 & 5 & 5 \\ 4 & 5 & 7 & 4 & 6 & 8 & 5 & 5 \end{bmatrix} \quad (2)$$

optimization degrees of freedom may result in an overall inferior final solution [DeS95].

We proved that the optimization problem of optimizing throughput is NP-complete when rephasing is considered simultaneously with algorithm/architecture matching by transforming the equal-subset problem into the new problem using the Karp's polynomial reduction technique. The proof is a variation of a similar proof for NP-completeness of throughput optimization using only algorithm/architecture matching [Pot94a].

When the goal is generation of the optimal solution of this problem one option is to build mathematical model which will serve as the input to the ILP formulation for solving throughput optimization using retiming or algorithm/architecture selection and retiming problems [Lei91, DeS95]. Using as a starting point the retiming algorithm [Lei91], first an important special case of the new problem when all computational blocks have an arbitrary number of inputs, but only 1 output can be solved. Once this special case is solved it can serve as a basis for solving the general problem, when no restriction is imposed on the number of outputs of the blocks. The detailed description of the mathematical model and the ILP-based solution is given in [DeS95]. However, it is important to note that already when only retiming is considered only relatively small instances of the general problem can be solved in reasonable run-times. As we already mentioned, most often in image and video processing considered computations are relatively large. In order to address the large instances of the problem, we developed the following, discrete relaxation-based, heuristic approach. While we from now on will consider only rephasing as transformation. The heuristic optimization algorithm can be easily retargeted to retiming and/or pipelining by replacing the modified Leiserson-Saxe algorithm in the 3rd step by the original Leiserson-Saxe algorithm for critical path minimization using retiming [Lei83].

Heuristic Algorithm for Throughput Optimization using Algorithm/Architecture Matching and Rephasing:

1. *Eliminate all inferior choices for each individual block;*
2. *Form a super-algorithm solution for each block;*
3. *Do optimal rephasing using the modified Leiserson-Saxe algorithm;*
4. *For each block substitute the super-algorithm solution with one of choices which.*

In the first step of the optimization algorithm all alternatives (matches between available algorithms and architectures) which have all distances between inputs and outputs longer than some other choice are eliminated without loss of optimality. In the second step of the heuristic algorithm super-algorithm is a new algorithm which has as the input-output distances the shortest distance between a given pair input-output as provided by any of the available algorithms. Note that after applying the third step of algorithm on this computational structure a lower bound on the final solution is obtained. The third step is direct application of the standard retiming for critical path minimization algorithm [Lei91] on the transformed CDFG where each delay can be positioned on arbitrary position inside any of the computational blocks. This is accomplished by replacing each computational block with unit delay blocks, while preserving the I/O distances required by the super-algorithms.

In the last step, the super-algorithms are replaced by the available choices, starting from the most critical blocks where the discrepancies between the super-algorithm and the available algorithms is the largest. After the each selection of a particular algorithm for the targeted computational block in the last step, the third step is repeated in order to better explore inter-dependencies between algorithm/architecture selection and rephasing.

3.0 Experimental Results

The following image and video processing examples have been optimized using the proposed methodologies and optimization algorithms.

(1) HOM - System for Homomorphic filtering of images [Rao90];

(2) IMAGE - System for Enhancement of compressed images [Rao90];

(3) LMS - Transform domain adaptive LMS filter [Rao90];

(4) DDPCM - Motion Compensation (MC) interframe DCT/DPCM coding with prediction [Jai81]; and

(5) DDCT - MC interframe DCT/DPCM coding with prediction in the temporal domain [Jai81].

Parameter Q in the LMS examples denotes different versions of updating algorithms with three different rates of convergence. The large values of Q correspond to a large number of states (and therefore large latency) in a feedback part of the computation.

Table 1 shows the obtained results achieved using the discrete-relaxation-based iterative heuristic optimization algorithm introduced in the previos section. This Table shows the initial throughput and the throughput after optimization using algorithm/architecture matching and rephasing The optimization is conducted by the heuristic algorithm introduced in the previous section. The average and median improvements are by factors of 2.65 and 2.42 respectively. It is interesting to note that although only heuristic solutions were obtainable, the improvements factors are higher than in the previously reported studies when the optimal, worst case exponential run-time, optimization algorithm which considers only retiming was used. This is so because the optimization power of rephasing at the system level is often significantly higher than the optimization power or retiming. Run times of the iterative heuristic algorithm were in all cases less than 30 seconds on SUN SparcStation IPX with 64 MB memory.

Design	Throughput before optimization	Throughput after optimization	Throughput Improvement
HOM	29	12	2.42
IMAGE	21	7	3.00
LMS (Q = 1)	32	18	1.78
LMS (Q = 2)	29	12	2.42
LMS (Q = 3)	21	10	2.10
DDPCM	39	11	3.55
DDCT	46	14	3.29

Table 1: Throughput optimization using algorithm selection and rephasing: Experimental Results

4.0 Conclusion

We studied the algorithm-architecture matching problem when simultaneously three powerful transformations, retiming, pipelining, and rephasing, are considered. After establishing the computational complexity of the optimization problem, we developed a simple and efficient iterative heuristic solution. The proposed discrete-relaxation-based heuristic solution solves large instances of the problem in very short run-times and achieves significant improvements over initial designs.

5.0 References

[Aiz95] K. Aizawa, T.S. Huang, "Model-Based Image Coding: Advanced Video Coding Techniques for Very Low Bit-Rate Applications", *Proc. of the IEEE*, Vol. 83, No. 2, pp. 259-271, 1995.

[Bar93] M.F. Barnsley, L.P. Hurd, "Fractal Image Compression", AK Peters, Wellesley, MA, 1993.

[Chi95] L. Chiariglione, "The Development of an Integrated Audiovisual Coding Standard: MPEG", *Proc. of the IEEE*, Vol. 83, No. 2, pp. 151-157, 1995.

[DeS95] J. DeSouza-Batista, M. Potkonjak, A. Parker, "Optimal Techniques for Video Algorithm/Architecture Matching and System Level Transformations,", NEC USA Technical Report, 1995.

[Don95] R.D. Dony, S. Haykin, "Neural Network Approaches to Image Compression", *Proc. of the IEEE*, Vol. 83, No. 2, pp. 288-303, 1995.

[Gup94] R.K. Gupta, C.N. Coehlo, G. De Micheli, "Program Implementation Schemes for Hardware-Software System", *IEEE Computer*, Vol. 27, No. 1, pp. 48-55, 1991

[Jai81] R. Jain, A.K. Jain, "Displacement measurement and its application ininterframe image coding", *IEEE Trans. on Comm.*, Vol. 29, No. 12, pp.1799-1808, 1981.

[Lee87] E. A. Lee and D. G. Messerschmitt, ``Static Scheduling of Synchronous Dataflow Programs for Digital Signal Processing", *IEEE Trans. on Computers*, Vol. 36, No. 1, pp. 24-35, 1987.

[Lee95] E.A. Lee, T.M. Parks, "Dataflow Process Networks", *Proc. of the IEEE*, Vol. 83, No. 5, pp. 773-799, 1995.

[Lei83] C.E. Leiserson, F.M. Rose, J.B. Saxe, "Optimizing synchronous circuits by retiming", *Proceedings of Third Conference on VLSI*, pp. 23-36, 1983.

[Ozc95] T. Ozcelik, J.C. Brailean, A. Katsaggelos, "Image and Video Compression Algorithms Based on Recovery Techniques Using Mean Filed Annealing", *Proc. of the IEEE*, Vol. 83, No. 2, pp. 304-316, 1995.

[Pir95] P. Pirsch, N. Demassieux, W. Gehrke, "VLSI Architectures for Video Compression - A Survey", *Proc. of the IEEE*, Vol. 83, No. 2, pp. 220-246, 1995 .

[Pot94a] M. Potkonjak, J. Rabaey, "Algorithm Selection: A Quantitative Computation-Intensive Optimization Approach", *ICCAD94 International Conference on Computer-Aided Design*, pp. 90-95, November 1994.

[Pot94b] M. Potkonjak, J. Rabaey, "Optimizing Resource Utilization Using Transformations", IEEE Transactions on CAD, Vol. 13, No. 3, pp. 277-292, March 1994.

[Pot95] M. Potkonjak, M.B. Srivastava, "Rephasing: A transformation technique for the manipulation of timing constraints", pp. 107-112, *Design Automation Conference, 1995*.

[Rao90] K.R. Rao, P. Yip:, "Discrete Cosine Transform", Academic Press, Inc., San Diego, CA , 1990

[Wol94] W.H. Wolf, "Hardware-Software Co-Design of Embedded Systems", *Proc. of the IEEE*, Vol. 82, No. 7, pp. 967-989, 1994.