From Algorithms to Hardware Architectures: A Comparison of Regular and Irregular Structured IDCT Algorithms

Claus Schneider , Martin Kayss , Thomas Hollstein[‡], Jürgen Deicke[‡]

Siemens Corporate Technology, ZT ME 5, 81730 Munich, Germany
E-Mail: claus.schneider@mchp.siemens.de

[‡] Darmstadt University of Technology, Institute of Microelectronic Systems, 64283 Darmstadt, Germany

Abstract

The inverse discrete cosine transformation (IDCT) is used in a variety of decoders (e.g. MPEG). On one hand, highly optimized algorithms that are characterized by an irregular structure and a minimum number of operations are known from software implementations. On the other hand, regular structured architectures are often used in hardware realizations. In this paper a comparison of regular and irregular structured IDCT algorithms for efficient hardware realization is presented. The irregular structured algorithms are discussed with main emphasis on assessment criteria for algorithm selection and high-level synthesis for hardware cost estimation.

1 Introduction

The two-dimensional discrete cosine transformation (2D-DCT) is used in many video coding standards (e.g. JPEG, MPEG-1/2) to reduce redundancy. Decorrelation is performed by transformation of image blocks (typically 8x8 pixels) into the frequency space. A decoder has to perform the inverse transformation, which can be separated into 2 one-dimensional inverse discrete cosine transformations (1D-IDCT). For two 1D-IDCTs, $2N^3$ multiplications and $2N^2(N-1)$ additions are needed to perform the matrix multiplications. That are 1024 multiplications and 896 additions for N = 8.

The number of operations can be reduced drastically by exploiting symmetries of the transformation matrix. The more optimizations are made to reduce the number of operations, the more irregular the algorithm gets. Therefore many hardware architectures are based on less optimized algorithms, that still have a regular structure. Using a multiply/accumulate (MAC) architecture, these regular structured algorithms can be designed with low effort. On the other hand, irregular structured algorithms with a minimum number of operations are known from software implementations. Same as the algorithm the hardware structure gets more and more irregular and therefore requires much effort for design.

For the transition from algorithms to hardware architectures, a trade-off between the number of operations and the regularity of the structure has to be made. The design flow and architecture trade-off process is demonstrated for both, a regular structured architecture, and an irregular structured one.

1.1 Related work

In literature, either application-specific hardware realizations [CPS89, RuTo92] or abstract descriptions of fast and highly optimized algorithms [CSF77, ArJu89, LLM89] can be found. The papers reporting on hardware realizations mainly focus on architecture details and mostly do not cover architecture exploration. Papers about algorithms are dominated by discussions about how to reduce the number of multiplications. A tool for editing, simulation and analysis of data flow graphs is presented in [LLM88].

But for the transition from algorithms to hardware architectures, many different architecture variants have to be examined, to select one for the final realization. In this paper regular and irregular structured algorithms are compared for efficient hardware realization, and assessment criteria needed for architecture selection are discussed.

1.2 Paper Structure

First, an overview of different IDCT algorithms and the architecture exploration process is given. After that, a regular structured hardware architecture based on MACs is presented. Then, irregular structured algorithms are discussed with main emphasis on assessment criteria for algorithm selection and behavioral synthesis for hardware cost estimation. A comparison of regular and irregular structured algorithms concludes the paper.

2 Overview

2.1 Discrete Cosine Transformation

The DCT was first introduced by Ahmed et al. in 1974 [ANR74]. In Equation 1, the direct algorithm to compute the 2D-DCT and in Equation 2 the inverse transformation is defined.

$$\frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \quad (1)$$

$$\frac{2}{N}\sum_{v=0}^{N-1}\sum_{u=0}^{N-1}C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \quad (2)$$

The two-dimensional DCT can be separated into a sequence of one-dimensional transformations. Therefore, 1D-DCTs are applied to all rows and all columns. The 1D-DCT is defined in Equation 3 and the inverse operation in Equation 4 using the constants of Equation 5.

$$X(m) = \sqrt{\frac{2}{N}} C(m) \sum_{k=0}^{N-1} x(k) \cos \frac{\Pi m(2k+1)}{2N} \quad (3)$$

$$x(k) = \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} X(m) C(m) \cos \frac{\Pi m(2k+1)}{2N} \quad (4)$$

$$C(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for i = 0} \\ 1 & \text{otherwise.} \end{cases} \quad (5)$$

Fast algorithms use a basic operation called rotation (see Equation 6). Graphically, the vector (x, y) is rotated by an angle ϕ to calculate the vector (X, Y).

$$\begin{aligned} X &= \cos(\phi) \cdot x + \sin(\phi) \cdot y = (\sin(\phi) - \cos(\phi)) \cdot y + t \\ Y &= -\sin(\phi) \cdot x + \cos(\phi) \cdot y = -(\sin(\phi) + \cos(\phi)) \cdot x + t \end{aligned} \quad (6)$$

The so-called fast rotation is obtained by calculating the common sub-expression $t = \cos(\phi) \cdot (x + y)$ first. At the cost of an additional register for t, the number of operations can be reduced from 4 multiplications and 2 additions to 3 multiplications and 3 additions.

2.1.1 Known Algorithms

In recent years, different algorithms have been published to compute the discrete cosine transform and its inverse. The goal of these approaches is the reduction of the number of multiplications. The lower bound for an 1D-DCT are 11 multiplications, as shown by Duhamel [DuMi87].

Matrix factorization is used by Chen et al. to half the computation costs of an N-point DCT, in which N is a power 2 [CSF77]. For an 8-point 1D-DCT, 16 multiplications and 26 additions(subtractions) are needed. The number of operations can be further reduced to 13 multiplications and 29 additions(subtractions) by applying the fast rotation to the algorithm.

Loeffler et al. introduced a version that requires only 11 multiplications and 29 additions(subtractions) for an 8-point DCT [LLM89] with a maximum path length of 2 multiplications and 4 additions.

Artieri et al. developed an algorithm that has a recursive structure and requires only 11 multiplications and 29 additions [ArJu89]. The drawback of this implementation is that the maximum data path contains up to three multiplications and four additions. The problem is the error propagation in the different stages, because of the necessity of rounding the results after the multiplications.

2.2 Architecture Exploration

First, regular and irregular structured algorithms are modeled in C to perform the IEEE compliance test. At this stage the bit widths of the intermediate results and thus for the operations have to be determined. For regular structured algorithms hardware costs can be estimated before RTL modeling and synthesis. On the other hand, for irregular structured algorithms, a VHDL behavioral model is developed. Resource estimation can be done with this model.

After selection of one promising algorithm, behavioral synthesis with different constraints is performed to explore the design space.

Finally, an architecture is selected for RTL synthesis.

3 Regular Architectures

Due to the relatively small modeling effort, many hardware realizations of the IDCT use regular structured algorithms. The transformation can be written in matrix form, as shown in Equation 7.

$$\underline{f} = \underset{\sim}{C}^T \cdot \underline{F} \quad (7)$$

For an 8-point 1D-IDCT the input vector F has to be multiplied with the transposed transformation matrix C to get the result vector f. Therefore, 8 x 8 multiply/accumulate operations (64 multiplications and 56 additions) are necessary. Because of the symmetry of the transformation matrix C, the transformation can be divided into two 4 x 4 matrix multiplications and a butterfly operation [CSF77], as shown in Equation 8 and Equation 9.

$$\underline{f}_{0,1,2,3} = \underset{\sim}{C}_1^T \cdot \underline{F}_{0,2,4,6} + \underset{\sim}{C}_2^T \cdot \underline{F}_{1,3,5,7} \quad (8)$$

$$\underline{f}_{7,6,5,4} = \underset{\sim}{C}_1^T \cdot \underline{F}_{0,2,4,6} - \underset{\sim}{C}_2^T \cdot \underline{F}_{1,3,5,7} \quad (9)$$

The even part of the input vector F is multiplied with the matrix C_1 and the odd with the matrix C_2. Intermediate results of these multiply/accumulate (MAC) operations are added and subtracted by a so-called butterfly operation to calculate the final results. A hardware architecture based on multiple MACs and one butterfly operation is shown in Figure 1.

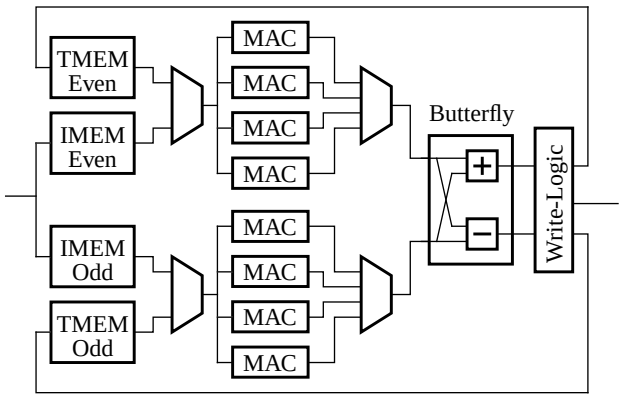

Figure 1: Regular Structured IDCT

Because of the separation of the 2D-IDCT into two one-dimensional ones, in the first step all rows and in the second step all columns of the matrix are processed, or vice versa. In the first step, the 1D-IDCTs are performed reading the input memories (IMEM) and writing the transpose memories (TMEM). The MAC operations can be performed in parallel for the even part and the odd part. For the butterfly operation the results from both parts are needed. In the second step, the intermediate results are read (TMEM) and the final results are written to the output.

The architecture shown in Figure 1 is scalable by the number of MACs. Even and odd part should consist of equal numbers of MACs to enable parallel operation of both parts and to avoid synchronization problems at the butterfly operation. The controller (counter and look-up table for multiplexor control signals) for the datapath can

be kept simple, if the total number of MACs is a power of two. Because of the regular structure, the hardware costs can be estimated rapidly.

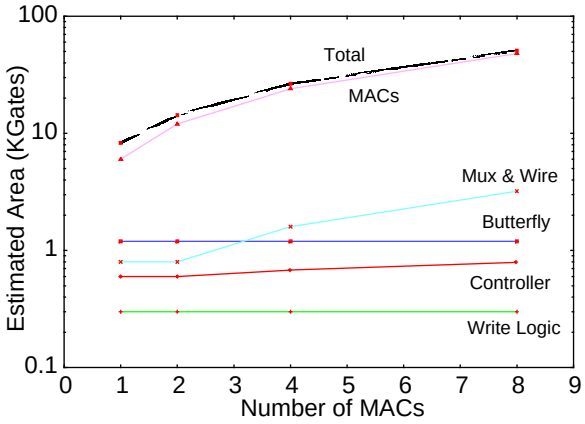

Figure 2: Hardware Cost Estimation

The overall circuit area is dominated by the area of the MAC operations, Figure 2. As the number of MACs increases, the multiplexor and wiring area is growing as well, but slower than the area of the MAC operations.

The regular structured approach is characterized by a simple datapath and controller, that can be modeled at RTL with relatively little effort. Hardware costs can be estimated rapidly before modeling and synthesis.

4 Irregular Architectures

4.1 Multiple Multiplications per Path

The algorithm proposed by Loeffler et al. [LLM89] requires only the theoretical minimum of 11 multiplications. In Figure 3, the irregular structured data flow graph (DFG) is shown.

But, besides the small number of operations, the algorithm has the drawback of multiple multiplications per data path. The multiplications are carried out with integer constants that are scaled to 12-14 bits depending on the precision requirements. The problem that arises for an area efficient hardware realization is the increasing bit width of the intermediate results and especially the resulting multiplier size. Only rounding can keep the intermediate results in a reasonable bit width. Because of the fixed-point arithmetic, this leads to an error propagation through the IDCT stages.

To solve this problem an architecture with only one multiplication per path has to be found.

4.2 One Multiplication per Path

The idea is to expand the number of operations in the odd part to achieve parallel multiplications. This can be done at the cost of one additional multiplication and three additions. Thus, the number of operations is 12 multiplications and 32 additions(subtractions). The advantage of this implementation is, that no data path contains more than one multiplication which allows an efficient and accurate hardware design. The resulting data flow graph is shown in Figure 4.

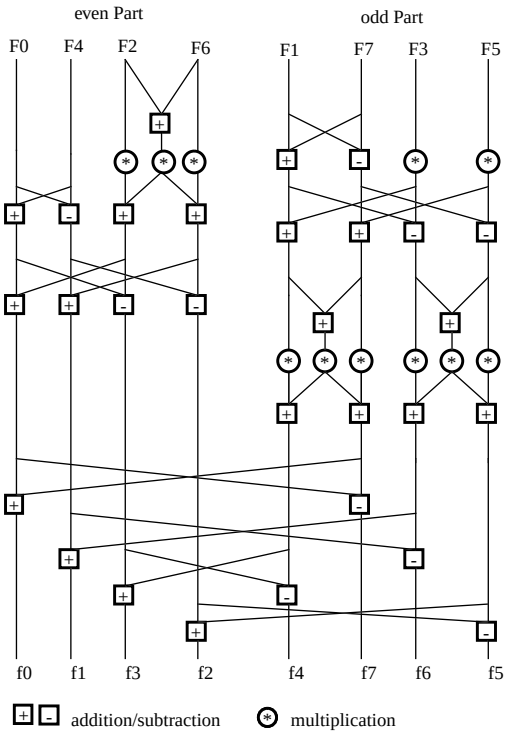

Figure 3: DFG (Loeffler) with 11 multiplications

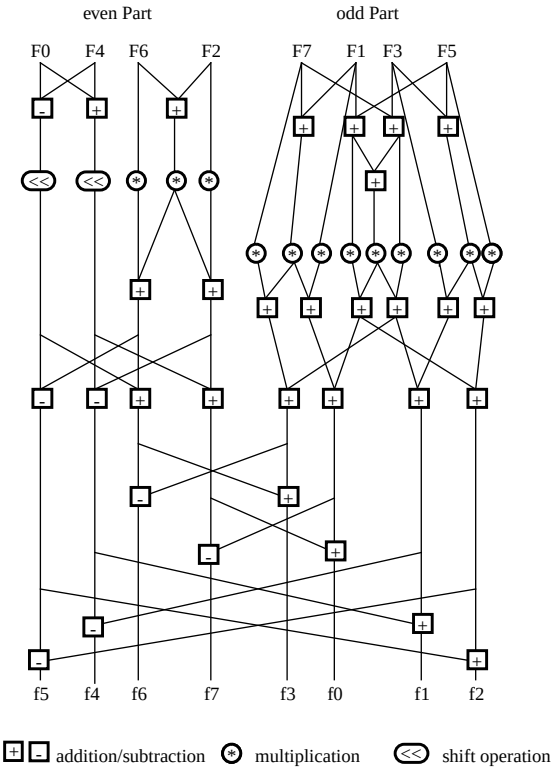

Figure 4: DFG (Loeffler) with 12 multiplications

4.3 Compliance Test

To guarantee consistency in the quality of the reconstructed values, a strict error limit is given for the MPEG standard. The different IDCT implementations use fixed-point arithmetic with a finite number of bits for the constants and the intermediate results. The input value to the IDCT is set to 12 bits and the output to 9 bits by the MPEG standard.

Because of the reduced precision due to the scaled constants and the rounded intermediate results, the IEEE1180-1990 compliance test [IEEE1180] is used to check the accuracy of the final results.

The test generates 60.000 data blocks, each consisting of 8 x 8 values, filled with random numbers. Compared to a floating-point implementation, the fixed-point IDCT algorithm has to reconstruct the blocks within a specified error limit.

Because of the high amount of data blocks the different algorithms were implemented as C software models to perform the compliance test. They were tested to meet the accuracy of the IEEE1180 with a minimum number of additional fractional bits. Therefore, the result of each multiplication and the output values after each 1D-IDCT were rounded. The rounding also scales the results to convert the values from fixed-point arithmetic.

The IEEE1180 compliance test is the crucial constraint for the selection of an algorithm for behavior synthesis.

4.4 Algorithm Selection

The different implementations of the inverse discrete cosine transform have to be evaluated by assessment criteria to find the best candidate for behavior synthesis. The number of resources is used as an indication to minimize the area of the resulting circuit.

Criteria		Chen	Loeffler11	Loeffler12	Artieri
#multiplications		13	11	12	11
#additions		16	20	25	17
#subtractions		13	9	7	12
#operations		42	40	44	40
path	#mult.	2	2	1	3
	#add.	5	4	4	4
#registers		9	12	11	10
#cycles		19	15	19	15

Table 1: Algorithm Selection Table

Based on the numbers in Table 1 an algorithm was selected for behavioral synthesis. The number of required registers was estimated by manual ASAP (As Soon As Possible) schedules and register lifetime tables. For the number of cycles, no I/O order and only one multiplication and two Add/Sub operations per cycle, were assumed.

The algorithm of Loeffler et al. with 12 multiplications was chosen. At the cost of one additional multiplication compared to the theoretical minimum, this implementation provides a simple solution for the problem of fixed-point arithmetic and passes the compliance test with good accuracy.

4.5 High-Level Synthesis

The usage of a behavioral description offers a lot of opportunities in comparison to the limited possibilities of RTL synthesis. Changes to the specifications are easier at behavioral level, than redesigning an RTL implementation. An introduction to high-level synthesis can be found in [GaRa94], while [MLD92] describes the high-level synthesis algorithms in more detail. Furthermore, an approach to exploring the algorithmic design space using high-level synthesis can be found in [PRE93].

The multiple options concerning the amount of resources and the timing were generated to determine an optimal architecture that can be synthesized.

The order of the input values at the two I/O ports is determined by local constraints because the IDCT demands none.

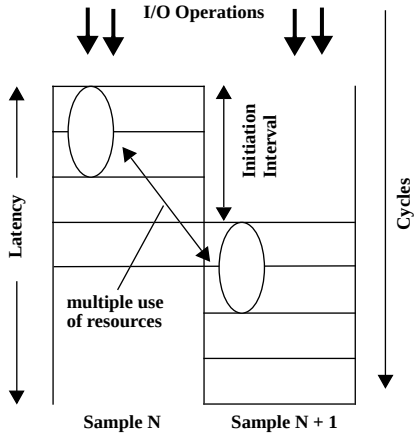

Figure 5: Initiation Interval and Latency

The usage of loop pipelining allows to increase the data throughput in user-defined steps. The length of the initiation interval (sample period) and the number of loops were varied to find an optimal architecture suitable for synthesis.

Different models with various options were generated to examine the search space for a suitable candidate for RTL synthesis.

4.5.1 Synthesis Results

The results of behavioral synthesis are shown in Table 2 and in Figure 6.

Initiation Interval	Latency	Loop	*	+/-	+	Inc./Dec.	Register (Bits)
5	20	4	3	8	1	0/0	572
6	18	3	2	7	1	2/2	352
11	22	2	2	4	0	0/0	340
12	24	2	1	4	0	0/0	388
20	20	1	1	3	0	0/0	284

Table 2: Resources (Behavior Level)

The number of resources in dependence of the length of the initiation interval is used to determine the optimal architecture for RTL synthesis.

Considering the results of behavioral synthesis there are two interesting boundaries in the search area. Between 5/6 and 11/12 cycles the area drops because the number of required multipliers decreases by one.

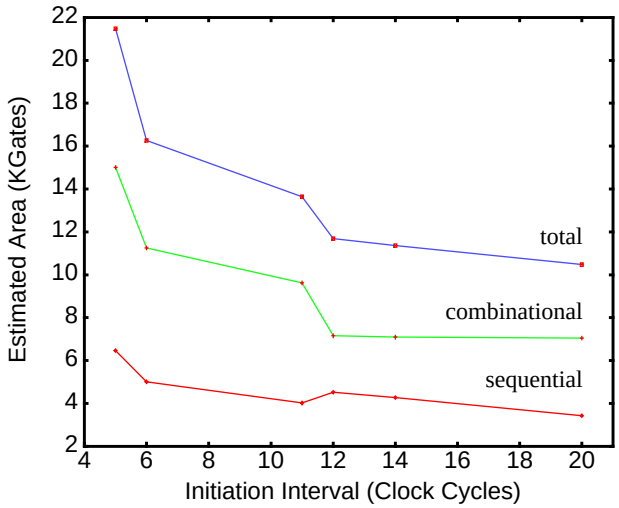

Figure 6: Area Estimation (Behavior Level)

The option with an initiation interval of 12 cycles and a latency of 24 cycles was chosen for RTL synthesis. The result of the synthesis with different timing margins showed that 46% of the total area is used by multiplexors and interconnect, 3% by the controller, 22% by registers and only 29% by the operations. The big multiplexor and wiring area compared to the relatively small area used by the arithmetic operations can be explained by the small number of operations and the high amount of operands.

5 Conclusion: Regular vs. Irregular

There is a big difference in development time between the two approaches. A long analysis phase is needed for the pre-selection of irregular structured algorithms for behavioral synthesis. Especially the process to determine the bit widths (to pass the compliance test) is more complex and time consuming for irregular structured algorithms, than for regular ones. This task can not be performed by high-level synthesis tools. But for design space explorations at behavior level, high-level synthesis is well suited to perform a trade-off between the number of cycles for the initiation interval on one hand, and the required number of resources on the other. In contrast to the high effort for architecture trade-off for irregular structured algorithms, hardware cost estimation can be done with low effort for regular structures.

However, the main advantage of irregular structured algorithms is the possibility to adjust the performance in a fine grain by the number of cycles of the initiation interval. The regular structured architecture can only be parameterized by the number of MAC operations. The number of MACs implies already the number of cycles (e.g. 1, 2, 4, 8 MACs correspond to 32, 16, 8, 4 cycles). The number of cycles of the behavioral description of the Loeffler algorithm can be defined in steps of one cycle by constraining the high-level synthesis. However, only a few combinations of the number of multipliers and the number of cycles result in efficient hardware realizations.

The transition from an optimized software algorithm to an efficient hardware architecture is difficult, because of the various requirements and the huge design space. A minimum number of operations is not as important for a hardware realization, as it is for a software implementation. On the contrary, because of the extensive resource sharing, multiplexors and wiring take nearly half of the total circuit area. This effect will more and more dominate in deep submicron, where wiring delay is getting more important than gate delay.

Acknowledgments

We would like to thank Sabine Rössel for all the helpful discussions and her support in modeling methodology and design constraining for behavioral synthesis.

References

[ANR74] Ahmed, N.; Natarajan, T.; Rao, K.R.: *Discrete Cosine Transform*. IEEE Transactions on Computer, January 1974, pp.90-93.

[ArJu89] Artieri, A.; Jutant, F.: *Procédé de détermination de transformée en cosine discrète*. Brevet Ndeg. 89 0234, 23. février 1989.

[CSF77] Chen, W.H.; Smith, C.H.; Fralick, S.C.: *A Fast Computational algorithm for the Discrete Cosine Transform*. IEEE Transactions on Communications, Vol. COM-25, No. 9, September 1977, pp. 1004-1009.

[CPS89] Carlach, J.C.; Penard, P.; Sicre, J.L.: *TCAD: A 27 MHz 8x8 Discrete Cosine Transform Chip*. IEEE Proceedings of the ICASSP, 1998.

[DuMi87] Duhamel, P.; H´Mida, H.: *New 2^n DCT Algorithms suitable for VLSI Implementation*. Proceedings IEEE International conference on Acoustics, Speech and Signal Processing, ICASSP-87, Dallas, April 1987, pp. 1805-1808.

[GaRa94] Gajski, D.D.; Ramachandran, L.: *Introduction to high-level synthesis*. IEEE Design & Test of Computers (1994) vol.11, no.4, p.44-54.

[IEEE1180] *IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform.* IEEEStd1180-1990, 1991.

[LLM88] Loeffler, C.; Ligtenberg, A.; Moschytz, G.S.: *Algorithm - Architecture Mapping for Custom DSP Chips.* IEEE Proceedings of the ISCAS, 1988.

[LLM89] Loeffler, C.; Ligtenberg, A.; Moschytz, G.S.: *Practical Fast 1D- DCT Algorithms with 11 multiplications*. IEEE Proc. Int´l Conf. on Acoustics, Speech and Signal Processing 1989 (ICASSP´89), pp. 988-991.

[MLD92] Michel, P.;Lauther, U.; Duzy, P.: *The Synthesis Approach to Digital System Design*, Chapter 6: *High-Level Synthesis*, by Sabine März. Kluwer Academic Publishers, 1992.

[PRE93] Potkonjak, M.; Rabaey, J.; ed. Eggermont, L.D.J.; Dewilde, P.; Deprettere, E.; Van Meerbergen, J.: *Exploring the algorithmic design space using high level synthesis*. VLSI Signal Processing, VI (Cat. No.93TH0533-0), IEEE: New York, NY, USA, 1993.

[RuTo92] Ruetz, P.A., Tong, P.; *A 160 MPixel/Sec IDCT Processor for HDTV.* IEEE Proceedings of the Custom Integrated Circuits Conference, 1992.