PIPELINE ARCHITECTURE FOR DCT/IDCT

Jari Nikara[1], *Jarmo Takala*[1], *David Akopian*[2], *and Jukka Saarinen*[1]

[1]Digital and Computer Systems Laboratory
Tampere University of Technology
P.O.B. 553, 33101 Tampere, Finland

[2]Nokia Mobile Phones
P.O.B. 429
33101 Tampere, Finland

ABSTRACT

In this paper, a unified sequential architecture for 8-point discrete cosine transform and its inverse is described. The architecture is based on rescheduled constant geometry algorithms, which are mapped onto a one-dimensional structure with the aid of vertical projection. The resulting modular architecture can be efficiently pipelined since arithmetic units are removed from the critical path. In addition, the complexity of the architecture is compared to other previously reported architectures operating over sequential data.

1. INTRODUCTION

The discrete cosine transform (DCT), considered as one of the best tools in digital signal processing, has many applications in the area of multimedia. Especially 8×8 DCT and its inverse (IDCT) have been popular in image compression applications. Real-time requirements of current applications often call for parallel implementations. In principle, parallel architectures can be developed by exploiting inherent spatial and/or temporal parallelism in fast algorithms for DCT and IDCT. However, such algorithms are often irregular, which may limit the exploitation level of the parallelism. In general, fast algorithms for DCT can be interpreted to contain consecutive processing stages composed of nodes of arithmetic operations. In order to minimize the cost of the implementation, the nodes should be mapped onto reduced number of processing elements and at the same time the throughput of the architecture should fulfil the given real-time requirements.

In principle, the exploitation of the spatial parallelism results in a column architecture where the computations are performed recursively on parallel data, i.e., nodes at a single processing stage are computed at a time. In such an architecture, the throughput is limited by the delay of basic arithmetic units used to realize the nodes. Exploitation of the temporal parallelism, in turn, results in sequential architectures, where the computations are performed over data in sequential form and the overall structure can be considered as a pipeline. In such an architecture, the throughput can be tailored with additional pipeline registers if data dependencies, i.e., feedback loops in the architecture, can be avoided. In addition, data is often in sequential form, thus architectures operating over sequential data are advantageous.

Several sequential architectures supporting DCT and IDCT have been proposed. In [1], a sequential architecture based on irregular algorithm from [2] is proposed. Due to the data dependencies, extensive pipelining is not possible and the high throughput in this architecture is obtained by utilizing several multiplier and accumulation units increasing the hardware cost. The pipelining efficiency can be improved by exploiting the temporal parallelism as used in architecture proposed in [3] where vertical projection (or folding) is utilized. This results in a partial-column architecture, i.e., the architecture exploits both temporal and spatial parallelism. In this architecture, the same arithmetic resources are used to compute both DCT and IDCT, which requires more complex control to reverse the signal flow.

The control complexity can be reduced by applying vertical projection to more regular DCT algorithm similar manner as, e.g., in architectural derivation of regular fast Fourier transform in [4]. Such an approach is used in sequential DCT/IDCT architecture proposed in [5]. The utilized fast algorithm is regular consisting of stages of Cooley-Tukey type of butterflies followed by irregular butterflies in DCT or preceeded in IDCT. The DCT and IDCT algorithms are rescheduled into a form where the topology of the regular kernel is identical. By collapsing the signal flow graph (SFG) onto a butterfly processing elements, a pipeline architecture is obtained. The architecture consists of sequential butterfly processors realizing the regular kernel, post-processor for realizing the irregular butterflies of the DCT algorithm, and preprocessor realizing the irregularities in the beginning of the IDCT algorithm. However, the coefficients of the used DCT algorithm are secants, which implies larger round-off errors compared to algorithms where cosines or sines are used as coefficients [6]. This, in turn, may increase the hardware cost if larger word width is needed in internal arithmetic for fulfilling the signal-to-noise requirements of a given application.

In this paper, a sequential architecture supporting DCT and IDCT is proposed. The architecture is based on the constant geometry DCT algorithm proposed in [7], which has a regular communication structure and all the coefficients are cosines. The architecture is illustrated with an example supporting 8-point DCT/IDCT. In addition, the complexity of proposed architecture is compared to other previously proposed sequential DCT/IDCT architectures.

2. FAST ALGORITHMS FOR DCT AND IDCT

The architectural derivation is based on the constant geometry fast algorithm for DCT of type II proposed in [7]. The SFGs of 8-point constant geometry forward and inverse DCT algorithms are depicted in Fig. 1 (a) and (b), respectively. The coefficients d_i can be generated recursively as

$$d_1 = \sqrt{0.5},\ d_{2i} = \sqrt{0.5(1+d_i)},\ d_{2i+1} = \sqrt{0.5(1-d_i)}.$$

These coefficients are actually cosines and will cause less round-off error compared to fast DCT algorithms where coefficients are based on secants [6]. The interconnection topologies of the SFGs

The first author acknowledges Nokia Foundation for financial support.

0-7803-6685-9/01/$10.00©2001 IEEE

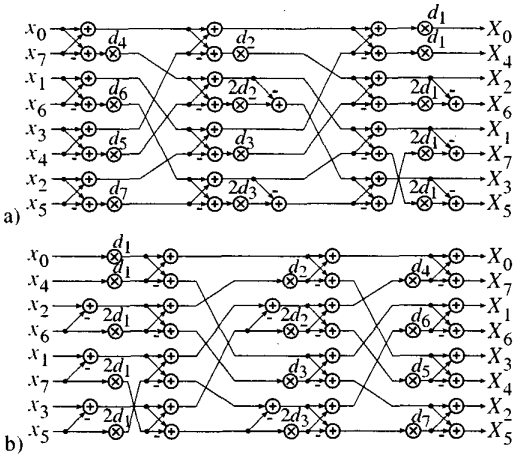

Figure 1: Signal flow graphs of constant geometry fast algorithm for (a) forward and (b) inverse 8-point DCT-II according to [7].

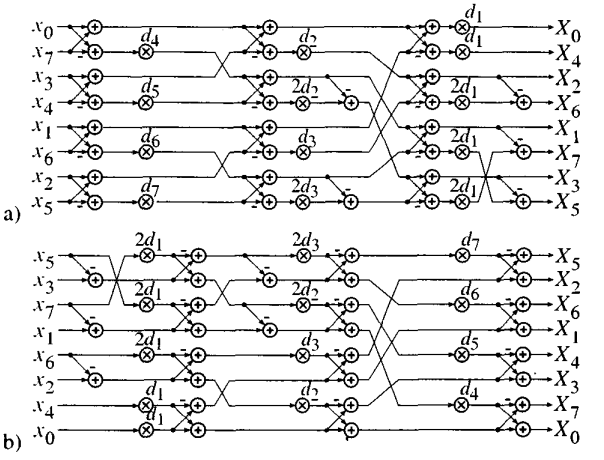

Figure 2: Signal flow graphs of rescheduled fast algorithms for (a) forward and (b) inverse 8-point DCT-II.

are regular, thus they lend themselves well to VLSI implementations. Therefore, the constant geometry algorithm is a good starting point for developing parallel architectures for DCT and IDCT.

The SFGs contain several similarities; in particular, all the reorderings between the processing stages are perfect shuffle permutations. Perfect shuffle permutation reorders elements of a sequence in such a way that the elements of the first half of a sequence are interlaced with the elements of the second half of the sequence, i.e., reordering of a vector $\mathbf{x} = (x_0, x_1, \ldots, x_{K-1})^T$ results in a vector $\mathbf{y} = (x_0, x_{K/2}, x_1, x_{K/2+1}, x_2, \ldots, x_{K-1})^T$.

A pipeline architecture supporting DCT and IDCT could be derived by collapsing both the SFGs in Fig. 1 and mapping them onto a common architecture. This approach is illustrated for IDCT already in [8]. However, the algorithms in this form will result in need for additional storage elements for realizing the constant size perfect shuffle permutations between the processing stages. In general, the minimum number of storage elements in the permutation network depends on the maximum distance a data element needs to be moved in the sequence. In perfect shuffle, the maximum distance in a 2^k-point sequence is $2^{k-1} - 1$. Therefore, for 2^k-point transform $(k-1)2^{k-1} - k + 1$ storage elements are needed for permutation networks.

The storage requirements of the reordering can be alleviated while maintaining the locality of the additional add operations by rescheduling the operations in SFGs in Fig. 1. Rescheduling should be performed in such a way that the interconnections will become perfect shuffle permutations with increasing size from stage to stage. Such a rescheduled versions of SFGs are illustrated in Fig. 2(a) and (b). It can be seen that the interconnections are still perfect shuffle permutations but with different sizes decreasing the total number of delay registers needed to realize the sequential reordering. The minimum storage requirement for realizing the reorderings in this algorithm for a 2^k-point DCT is $\sum_{i=1}^{k-1}(2^i - 1) = 2 \cdot 2^{k-1} - k - 1$. Furthermore, it can be seen that the topologies of interconnections in DCT in Fig. 2(a) and IDCT in Fig. 2(b) are similar, which implies that the corresponding reorderings in the algorithms can easily be mapped onto the same structures.

3. ARCHITECTURE

In principle, the SFGs in Fig. 2 can be interpreted to contain stages of parallel arithmetic operations and reorderings. Our purpose is to reduce the dimensionality of the SFG applying vertical projection, i.e., the SFGs are collapsed or folded into a one-dimensional SFG. Such a one-dimensional graph implies a sequential implementation.

3.1. Data Processing

The data processing in the SFGs can be considered to contain butterfly operations, multiplications and local subtractions. Vertical projection of the SFGs implies that each parallel stage of multiplications is mapped onto a single multiplier. Since the data is in sequential form, each data element is passed through the multiplier allowing efficient utilization of the numeric range of fixed-point systems. Therefore, the intermediate signal values can be scaled without additional hardware cost implying less round-off noise.

An efficient sequential realization for the parallel butterflies from area point of view is to use only one arithmetic unit, which can be controlled to perform either addition or subtraction as described in [5]. In such an arrangement, additional delay registers are needed since both the operands should be available for two sample periods although they enter at consecutive clock periods. The sequential structure based on adder/subtractor realizing the butterfly operation in Fig. 3(a) is illustrated in Fig. 3(b). The adder/subtractor in this structure is fully utilized at the expense of additional delay registers. Nevertheless, the number of delay registers is minimized in butterfly elements.

The SFGs in Fig. 2 contains additional subtractions shown in Fig. 3(c), which are always performed over consecutive data samples in the sequence, i.e, the subtractions are local. Such an operation in sequential form can be realized by delaying the preceding sample and performing the subtraction on the sample pair. However, the subtraction is not performed for all the sample pairs, thus an additional multiplexer is needed to bypass the local subtractor unit as illustrated in Fig. 3(d). In addition, by noting that the local

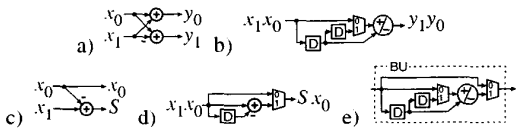

Figure 3: Mapping of basic operations onto sequential realizations: (a) signal flow graph of butterfly operation and (b) corresponding sequential realization, (c) signal flow graph of local subtraction and (d) corresponding sequential realization, and (e) sequential butterfly unit realizing both the previous operations. D: Delay register. BU: Butterfly unit. Clock signals omitted for clarity.

subtraction can be interpreted as a half of the butterfly operation, the local subtraction can be realized with the previous butterfly unit with an additional multiplexer to bypass the arithmetic operation. Therefore, a multi-function arithmetic unit realizing the butterfly operations and local subtractions in sequential form is a butterfly unit (BU) shown in Fig. 3(e). With the aid of this unit and a multiplier, all the data processing needed in the DCT and IDCT algorithms in Fig. 2 can be realized.

3.2. Data Reordering

The global reorderings between the processing stages in the algorithms in Fig. 2 are based on perfect shuffle permutations. In both SFGs, the first interconnection contains two 4-point perfect shuffles and the second is a single 8-point perfect shuffle. The perfect shuffle permutation in sequential form can be realized with the aid of sequential permutation networks proposed in [9] consisting of cascaded shift-exchange units (SEU) illustrated in Fig. 4(a). The SEU of size K contains a K-stage shift register with feedback and, in principle, it exchanges data elements in a sequential sequence K elements apart. Perfect shuffle permutation of a 2^k-point sequence can be performed with a structure where $k-1$ SEUs are cascaded in decreasing order of size: $\mathrm{SEU}_{2^{k-2}}$, $\mathrm{SEU}_{2^{k-3}}$, ..., SEU_{2^0}. The block diagram and principal operation of such a network for an 8-point perfect shuffle is illustrated in Fig. 4(b) and (c). It should be noted that a single SEU_1 performs the 4-point perfect shuffle permutation. Therefore, the global reorderings found in the SFGs can be realized with the structures shown in Fig. 4.

Besides the global reorderings between the processing stages, the rescheduled algorithms contain also local reorderings, which require data elements two elements apart to be exchanged before the last subtraction operation in the DCT or before the first multiplication in the IDCT as seen in the SFGs in Fig. 2. This exchange can be realized with a SEU unit containing a two-stage shift register, i.e., SEU_2. The exchange operation is performed only for a particular pair of data elements, which can easily be arranged by providing an appropriate control signal for the SEU_2 unit. In addition, it should be noted that, in these algorithms, the distance of the elements to be exchanged locally is constant two regardless of the transform size [7]. In cases where inorder inputs and outputs are needed, additional sequential permutation networks can be utilized as described [9].

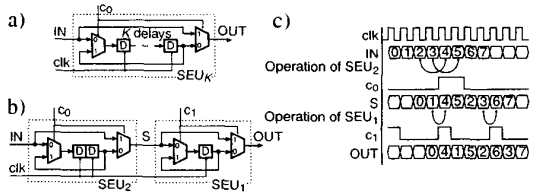

Figure 4: Sequential permutation: (a) shift-exchange unit (SEU) of size K, (b) block diagram of 8-point perfect shuffle permutation, and (c) corresponding timing diagram. c_k: control signal.

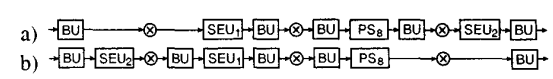

Figure 5: Block diagrams of sequential architecture for (a) DCT corresponding Fig. 1(a) and (b) IDCT corresponding Fig. 1(b). BU: Butterfly unit. SEU_K: Shift-exchange unit of size K. PS_8: 8-point perfect shuffle network.

3.3. Unified DCT/IDCT Architecture

According to the previous discussion, the final architectures for the DCT and the IDCT can be constructed by cascading the basic processing and reordering units, which were obtained by applying vertical projection to the operational stages of the SFGs of the rescheduled constant geometry DCT algorithms. Sequential architectures for the 8-point DCT and IDCT, based on the SFG in Fig. 2(a) and (b), are illustrated in Fig. 5(a) and (b), respectively. Each unit in the architecture corresponds to a specific operational stage in the SFG of the algorithm.

The architectures in Fig. 5 imply that a unified architecture supporting both DCT and IDCT can be constructed by providing additional data paths to reverse the data flow of the DCT pipeline for IDCT computation. In other words, the functional units of the DCT pipeline in Fig. 5(a) are arranged in reversed order for IDCT computation as described, e.g., in [3]. Such an approach will, however, introduce high routing costs and complicated control. Therefore, the data flow direction through the architecture should be the same in both transforms.

A more efficient solution is obvious when comparing the structures in Fig. 5: DCT and IDCT pipelines can be mapped onto a unified processing pipeline. Such an architecture for 8-point DCT and IDCT is illustrated in Fig. 6. According to SFG in Fig. 2(a), DCT can be computed by using the first BU for butterfly operation, bypassing the first SEU_2 unit and the second butterfly unit. The SEU_1 unit is used to realize the 4-point perfect shuffle. The third and fifth BUs are used for butterfly operation, while fourth and sixth BUs are for local subtractions. 8-point IDCT, in turn, can be computed by bypassing the last SEU_2 unit and the second to last butterfly unit and configuring the other BUs according to the SFG in Fig. 2(b).

The minimum latency of the DCT or IDCT computation is 11 clock cycles, which is only due to the data reordering. Assuming that each arithmetic unit is followed by a pipeline register, the minimum latency is 19 cycles. It should be noted that such pipeline registers are not included in Fig. 6. The unified architecture can be freely pipelined since there are no feedback loops in the structure. However, additional pipelining increases the latency although it improves the throughput.

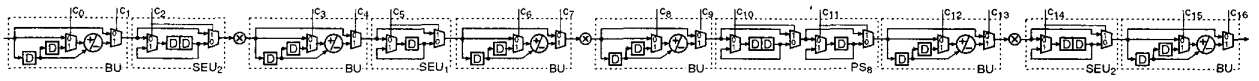

Figure 6: Block diagram of one-dimensional 8-point DCT/IDCT architecture. Clock signals are omitted for clarity.

4. COMPARISON

In order to compare the proposed architecture to other, previously proposed DCT/IDCT architectures, we have estimated their complexity based on the number of arithmetic units, multiplexers, and delay registers. The number of multiplexers and registers describes mainly the complexity of data reordering. The number of multiplexers is estimated as equivalent 2-to-1 multiplexers. We have also included pipeline registers after each arithmetic unit. The principal complexity estimates of different architectures for the 8-point DCT/IDCT are listed in Table 1.

In the architecture proposed by Madisetti and Wilson [1], the fact that the used DCT algorithm contains only seven different coefficients is utilized in such a way that an optimized, hard-wired multiplier is dedicated for each coefficient. The irregularity of the algorithm reflects the number of registers and multiplexers. The architecture by Cheng *et al.* [3] consists of two separate processor units and the control unit, which takes care of communication. Parallelism makes the high speed operation possible but introduces also high number of arithmetic units. Moreover, the control unit includes several multiplexers since the inverse DCT is performed by reversing the data flow. The architecture by Hsiao *et al.* [5] is based on an in-place algorithm, which reflects the larger number of registers than in the proposed architecture. The computational complexity of the architecture is comparable with the proposed architecture. The kernel operation consists of regular butterflies, thus additional pre-processing for DCT and post-processing for IDCT is needed. Furthermore, the architecture can be freely pipelined, thus the throughput rate is comparable with the proposed architecture.

5. CONCLUSION

In this paper, a unified fully sequential architecture for 8-point DCT and IDCT has been described. The one-dimensional architectures for separate DCT and IDCT are derived by applying vertical projection to DCT and IDCT algorithms, which are based on constant geometry fast algorithm for DCT-II. Finally, the architectures are combined as a unified pipeline architecture, which is able to perform the DCT or the IDCT. This introduces additional resources, one SEU_2 and BU, compared to single transform architecture but minimizes the routing. The combined architecture yields the complexity of 3 multipliers, 6 adder/subtractors. In principle, the proposed architecture can be extended to support 2^k-point DCT/IDCT by utilizing the approach described in this paper. The architecture is modular; each additional power of two in transform size requires addition of perfect shuffle network for global reordering, SEU unit for local reordering and two butterfly units for butterfly operations and local subtractions.

Table 1: Complexity comparison of DCT/IDCT architectures. M: Number of 2-to-1 multiplexers. D: Number of delay registers. $\times$: Number of multipliers. $+$: Number of adders. $+/-$: Number of add/subtract units.

Parameter / Reference	M	D	$\times$	$+$	$+/-$
[1]	≥ 42	≥ 20	7	2	8
[3]	several	unknown	8	-	21
[5]	8	34	3	4	3
Proposed	22	29	3	-	6

6. REFERENCES

[1] A. Madisetti and A. N. Wilson, "A 100 MHz 2-D 8×8 DCT/IDCT processor for HDTV applications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 5, no. 2, pp. 158–165, Apr. 1995.

[2] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. on Communications*, vol. 25, no. 9, pp. 1004–1009, Sept. 1977.

[3] K.-H. Cheng, C.-S. Huang, and C.-P. Lin, "The design and implementation of DCT/IDCT chip with novel architecture," in *Proc. IEEE Int. Symposium on Circuits and Systems*, Geneva, Switzerland, May 28–31 2000, pp. 741–744.

[4] H. L. Groginsky and G. A. Works, "A pipeline fast Fourier transform," *IEEE Trans. on Computers*, vol. 19, no. 11, pp. 1015–1019, Nov. 1970.

[5] S.-F. Hsiao, W.-R. Shiue, and J.-M. Tseng, "A cost efficient fully-pipelinable architecture for DCT/IDCT," *IEEE Trans. on Consumer Electronics*, vol. 45, no. 3, pp. 515–525, Aug. 1999.

[6] Z. Wang, "Pruning the fast discrete cosine transform," *IEEE Trans. on Communications*, vol. 39, no. 5, pp. 640–643, May 1991.

[7] J. Takala, D. Akopian, J. Astola, and J. Saarinen, "Constant geometry algorithm for discrete cosine transform," *IEEE Trans. on Signal Processing*, vol. 48, no. 6, pp. 1840–1843, June 2000.

[8] J. Nikara, J. Takala, D. Akopian, J. Astola, and J. Saarinen, "Pipelined architecture for inverse discrete cosine transform," in *Proc. X European Signal Processing Conference*, Tampere, Finland, Sept. 4–9 2000, vol. I, pp. 279–282.

[9] C. B. Shung, H.-D. Lin, R. Cybher, P. H. Siegel, and H. K. Thapar, "Area-efficient architectures for Viterbi algorithm I. Theory," *IEEE Trans. on Communications*, vol. 41, no. 4, pp. 636–644, Apr. 1993.