Design of a 2D DCT/IDCT application specific VLIW processor supporting scaled and sub-sampled blocks

Rohini Krishnan, O.P.Gangwal, Jos.v.Eijndhoven,
Philips Research Laboratories,
Eindhoven, The Netherlands.
Email : rohini.krishnan@philips.com.

Anshul Kumar,
Dept. Of Computer Science,
Indian Institute Of Technology,
New Delhi.

Abstract

We present an innovative design of an accurate, 2D DCT IDCT processor, which handles scaled and sub-sampled input blocks efficiently. In the IDCT mode, the latency of the processor scales with the size of the input blocks varying from 7 cycles for an 1x1 block to 38 cycles for an 8x8 block. This scalability is possible because the processor has input data dependant control by which it can exploit the reduced computational needs of sub-sampled blocks and blocks of smaller sizes to work in lesser cycles. This is a very useful feature for MPEG and HDTV decoders and has hitherto not been exploited. Clocking at 150 Mhz, the processor satisfies the high sample rate requirement of dual MPEG stream HD decoding with a picture size of 1920 x 1080 at 30 frames per second. Fixed word length and accuracy simulations of our design shows that it conforms to the accuracy specifications of the CCITT standard within a 16 bit data path.

A methodology based on architecture level synthesis is used to design the VLIW processor core. The VLIW design exploits the Instruction Level Parallelism present in the DCT/IDCT application, efficiently. The processor core is characterised by an area of 0.834 mm sq. and a frequency of 150 Mhz in 0.18 micron CMOS technology.

1 Introduction

The Discrete Cosine Transform (DCT) is a compute intensive part of the encoders and decoders of JPEG, MPEG, digital HDTV etc. In the past, several architectures for DCT/IDCT processors for various application domains based on different algorithms have been proposed [12]. All the previous architectures have been optimised for either performing IDCT on fixed 8x8 blocks [1] typically used in decoders (MPEG, JPEG etc.), or performing both FDCT and IDCT on fixed 8x8 blocks [2]-[6]. None of these processors are capable of exploiting the reduced computational needs of scaled or sub-sampled blocks to give a higher throughput. Almost all the hitherto published architectures have a fixed latency for all blocks. Paek et al [7] support a dual mode of operating on 8x8 blocks or 2x4x8 blocks for the digital VCR domain, thus providing support for scaled blocks of size 4x8. In our work, we go a step further by providing full support for scaled and sub-sampled blocks of sizes (1,2,4,8)x(1,2,4,8). Furthermore, the fixed point accuracy of our design meets the specifications of the CCITT[10] which is more stringent than that of the digital VCR domain.

Scaled blocks (of sizes less than 8x8) can arise in MPEG-2 decoders. For instance, the output of the inverse quantiser in a MPEG-2 decoder, can have many rows with zero valued coefficients. This output serves as an input to the inverse DCT processor. Instead of communicating the the zero valued rows, only the non-zero rows can be communicated from the output of the inverse quantiser to the inverse DCT processor. This would not only reduce the communication between the inverse quantiser and the inverse DCT processor but also speed up the decoding due to the reduced computational requirements. Sub-sampled blocks are those in which components of certain frequencies are omitted. Sub-sampled blocks can arise, for example, due to Picture In Picture requirements of HDTV where one picture is displayed inside another, at the cost of a reduced resolution of one. Another application where sub-sampled blocks can arise is in High Definition to Standard Definition down-conversion. The speed up in decoding can only be achieved if the inverse DCT processor is capable of working at a reduced latency for scaled and sub-sampled blocks.

In summary, the proposed architecture can perform FDCT and IDCT and in the IDCT mode support sub-sampled and scaled blocks. It is area-efficient and uses a common 16 bit data path for performing both FDCT and IDCT. It shows excellent scaling in its performance by having a higher throughput for sub-sampled and scaled blocks. The latency of the processor varies from 7 cycles to 38 cycles, depending on the input block size. It works at 150

0-7695-1868-0/03/$17.00 (C) 2003 IEEE

Mhz within a latency of 50 cycles and can hence handle dual MPEG HD stream. The VLIW architecture fully exploits the instruction level parallelism (ILP) present in the DCT/IDCT application. It conforms to the accuracy requirements of CCITT[10]. Our approach uses the high level synthesis methodology offered by A|RT designer[9]. By working at a high level of abstraction, we explore a larger design space, evaluate the effect of each design decision on the area-time constraints early in the design cycle, and perform test simulations and functional verification using the high level C/C++ language. We thus arrive at a feature-rich processor in a short design time.

2 Algorithm design and fixed point performance

Many fast DCT algorithms have been published in literature[12]. We consider a solution based on the fast algorithm proposed by Loeffler *et al.* [8], popularly known as the LLM algorithm. For sub-sampled input data some interesting features of the LLM Signal Flow Graph(SFG) can be exploited for achieving excellent scaling performance. Moreover, the LLM algorithm has the least number of multiplications for non-scaled data output. In addition, the SFG of LLM for forward and inverse DCTs are mirror images of one another and hence we can re-use the entire datapath for performing the forward and inverse transforms.

2.1 Subsampled block support and accuracy

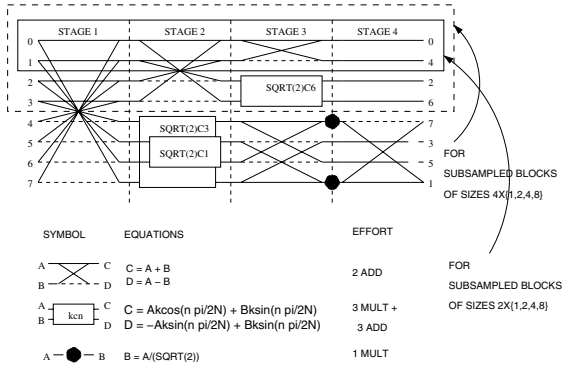

Figure 1. *SFG for LLM algorithm.*

As can be seen from Fig.1, for subsampled blocks of sizes 4x(1,2,4,8), we perform only the upper half computation(as indicated by the dashed line box) and for subsampled blocks of sizes 2x(1,2,4,8) we perform only the upper quarter computation(as indicated by the solid-line box). This is because the other coefficients are zero. By exploiting this property, we achieve excellent scalability in the latency and a higher throughput for sub-sampled blocks.

Table 1. *Fixed Point Performance of LLM, PPE=Pixel Peak Error, PMSE=Pixel Mean Sq. Error, OMSE=Overall Mean Sq. Error, PME=Pixel Mean Error, OME=Overall Mean Error*

Measure	Threshold for IDCT	Error for data range -256:255	Error for data range -5:5	Error for data range -300:300
PPE	1	1	1	1
PMSE	0.060	0.018	0.018	0.016
OMSE	0.020	0.015	0.015	0.013
PME	0.015	0.003	0.004	0.003
OME	0.0015	-0.0002	0.0001	0.0002

We use the following three techniques to reduce fixed point errors, prevent overflow and conform to the required accuracy within a 16 bit data path. Firstly, we modify the original signal flow graph in [8] and place the $\sqrt{2}$ factors, as shown in the signal flow graph in Fig.1. By modifying the original signal flow graph, such that there is no multiplication by a constant greater than 1, no overflow occurs at any intermediate stage, and so we avoid downscaling of data which could introduce a loss of accuracy. Secondly, we use 12 bit signed input for forward DCT, 16 bit signed input for Inverse DCT and 16 bit unsigned cosine constants. Thirdly we use a high precision rotator with an internal 32 bit adder, as explained in the next section. We performed C simulations to determine the fixed point performance of our solution approach based on the LLM algorithm. The results are as shown in Table.1 for three ranges of data as specified in CCITT [10].

3 Design of 2D DCT/IDCT processor

We obtain a worst case estimate of the speed at which the DCT/IDCT Processor should work to satisfy the requirements of dual stream HD MPEG-2 decoding. In the 1080i high definition format, the picture is 1920 x 1080 pixels, sent at 30 complete frames per second. A Macro Block is of size 16 x 16 pixels. The number of Macro Blocks per frame is thus $\frac{1920X1080}{2^8}$ which is approximately 8K macroblocks per frame. Assuming 4:2:0 format 8K macroblocks = 8K x 6 = 48K blocks/frame. At 30 frames per second, this translates to 48K x 30 which is approximately 1.5M blocks/sec. At 150 MHz as specified in CCITT [10], it means that the DCT/IDCT Co processor gets, for a dual HD stream $\frac{150MHz}{2X1.5Mblocks/sec}$ = 50 cycles per block of data.

3.1 Design methodology

We briefly explain the behavioural synthesis methodology based on which we design the processor core and

achieve the support for scaled and sub-sampled blocks. The A|RT Designer tool [9], which we use for the design of our processor core, takes as input the behavioural description of the algorithm in a subset of C, and generates a RT level synthesisable HDL model of the application specific VLIW processor. The classical design flow steps begins with the compilation of the source code inclusive of identifying and accurately representing the parallelism present in the source program. This is followed by creation of an architecture and mapping the source code data flow to the architecture. Removal of redundant hardware, scheduling of the operations and finally generating the microcode control ROM and RT level synthesisable HDL model completes the design flow steps shown in Fig.2.

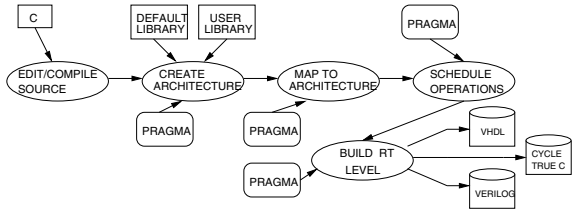

Figure 2. *A|RT design flow steps.*

In addition to the default library present in the tool framework (consisting of ALU, MAC, Multiplier etc), the designer can describe a custom library, where the resource can be modelled as a black box with only the time shape of the inputs and outputs described. The detailed HDL or cycle true C model of the custom components needs to be provided only before the RT model of the processor is built. The mapping of the operations to the type of functional unit is done by the designer during the mapping to architecture phase. The schedule step generates the total cycle count for the particular combination of resources and mapping. The A|RT tool generates the VLIW controller (consisting of micro code ROM, status and branch logic blocks, program counter), register files and multiplexers. The inputs of the controller are the status flags generated in the data path as well as a reset, initialise, and start up signal generated by the external hardware. The controller uses them to determine which instructions to execute next.

3.2 Design of computational core

The butterfly operation in Fig.1 translates to two additions in terms of effort. The *butterfly* is modelled as a single cycle resource. It takes two 16 bits normalised signed inputs and produces two 16 bits normalised signed outputs, one being the sum of the two inputs, the other being the difference.

The rotation operation is a dual output one. It is implemented using two single output resources as shown in Fig.3.

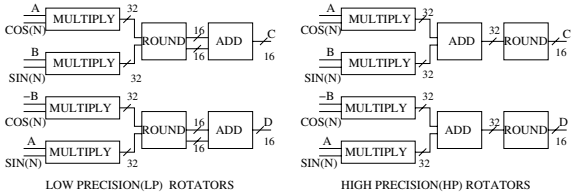

Figure 3. *Block diagram of LP and HP rotators.*

Table 2. *Fixed Point Performance for FDCT with HP and LP rotator.*

Measure	Data range (-256:255) HP, LP	Data range (-5:5) HP, LP	Data range (-300:300) HP, LP
PPE	1, 1	1, 1	1, 1
PMSE	0.126, 0.146	0.122, 0.148	0.124, 0.149
OMSE	0.101, 0.114	0.100, 0.119	0.101, 0.122
PME	0.009, 0.012	0.009, 0.013	0.009, 0.010
OME	0.0006, 0.0001	0.0002, 0.0002	0.0004, 0.0007

The *rotator* is a two cycle pipelined resource. It consists of a multiplication followed by an addition and rounding stage as shown in Fig.3. The rotator can be designed as a high precision (HP) rotator or a low precision (LP) rotator. The low precision version, has a multiplication stage followed by rounding and 16 bit addition. The high precision version, has a multiplication stage followed by 32 bit addition and rounding.

Table.2 shows the improved accuracy achieved in HP rotator by rounding after the 32 bit addition and thereby reducing the frequency of the last bit in error. Table.2 only shows the fixed point simulations for the forward DCT, since the accuracy for inverse DCT is acceptable with both the LP and HP rotators. It can be seen from Table.2 that for the forward DCT, using a LP rotator, the mean square errors are significantly higher than when a HP rotator is used. Since the 32 bit output from the adder is rounded by the rounding unit to produce a 16 bit output, the external interface remains at 16 bits. If a lower accuracy version can be tolerated, LP rotators can be used. Throughout this discussion, when we say rotator, we refer to the single output resource, as shown in the upper half or lower half of the block diagram for high precision (HP) rotators. So, a rotation operation is performed by two single output rotators.

Any good logic synthesis tool would produce a compact multiplier when either of the inputs to the multiplier is a constant (resulting in a hardwired, constant coefficient multiplier) [11]. In our implementation, we re-use the constants of the same precision for both FDCT and IDCT and thus re-use the compact rotator blocks for both the forward and inverse operations. From the LLM signal flow graph it can

be seen that there are 3 Rotation operations, each with a different constant and 2 multiplications with $1/\sqrt{2}$ (which can be considered as a rotation by 45 degrees with one of the inputs to the rotator being zero).

The *Logic Unit* (LU) is a single cycle combinational unit which needs the following inputs
(a) Scale/No Scale
(b) Number of rows in input block
(c) Horizontal Scale Factor (HSF)
(d) Vertical Scale Factor (VSF)
(e) Fdct or Idct. The logic unit produces twenty, 1 bit outputs, in 1 cycle, each output indicating a particular control condition. The presence of the logic unit eliminates the overhead for control condition checks.

The *Round and Saturation unit* is a single cycle unit. It gets eight, 16 bit numbers as inputs. It is a multi mode resource. In the FDCT case, it produces eight, 12 bit rounded and saturated outputs. In the IDCT case, it produces eight, 9 bit rounded and saturated outputs. In the design of this processor all the intermediate values are stored in registers and the design of a transposition memory is avoided.

3.3 Scaled and sub-sampled block support

The scalability in performance is achieved by avoiding redundant calculations. For eg., bit 2 output of the LU indicates that the input block is vertically subsampled by 2 and is of size 8x4, bit 17 indicates that the input block is scaled and is of size 4x8 etc. In a vertically subsampled block of size 8x4, only the even elements are non zero and thus the odd half of the IDCT flow graph can be bypassed (in effect performing a 4 point IDCT). In a scaled 4x8 block, we perform only 4 row-wise and 8 column-wise transforms.

3.4 Design space explorations

In the A|RT design methodology, the user has to manually map the operations to the resources. The number of resources of each type also needs to be decided by the user. To determine the number of resources of each type a good area-time tradeoff point needs to be found. We experimented with various combinations of general purpose and custom resources. The general purpose resources are available to the designer in the default library of the tool. For the custom resources, the designer needs to define the time shape of the inputs and the outputs.

General purpose/custom units can be used for performing the data path operations. For eg., the butterfly consists of one addition and one subtraction operation. It can be implemented by mapping these operations to the arithmetic and logic units (ALUs) present in the default library. To perform the addition and subtraction in parallel, each butterfly has to be mapped onto two ALUs. This mapping for each of the butterfly operations has to be done manually in A|RT, which is a bit tricky, since a bad mapping could result in a bad schedule and cycle overheads. The mapping is much simpler when we use custom units for the operation, since there is a direct correspondence between the operation and the functionality of the custom unit. We evaluated the cycle and regfile size overhead for each combination of general purpose and custom units. This is illustrated in Fig.4. Ap-

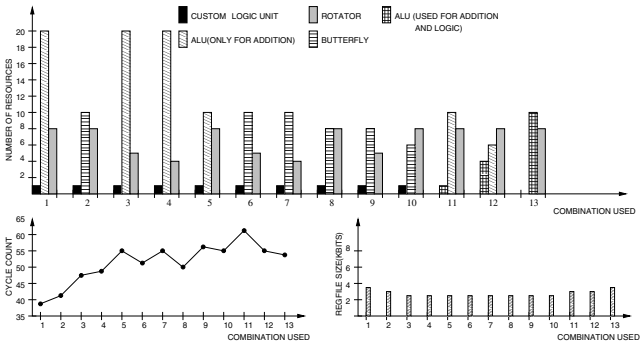

Figure 4. *Cycle Count for various combinations of resources.*

proach 1 with 20 ALUs, 8 rotators, 1 custom logic unit has the best cycle count of 38, but the area overhead is high. The register file size is 3.8 kbits. Approach 2 with 10 Butterflies, 8 rotators, 1 custom logic unit has a cycle count of 41 and a register file size of 3 kbits. In Approach 11, we map the logic operation of the control condition check and addition of the butterfly to the same ALU, and observe the cycle count and register file size. In some of the approaches, we use dedicated ALUs for addition (i.e. logic operations are not mapped on the same ALU).

We conclude that Approach 2 is promising and provides the most optimal area-cycle count trade off point. In Fig.4 we do not show the rounding and saturation unit, input and output ports which are common to all the approaches. We use 1 rounding and saturation unit and 8 input ports and 8 output ports in all the approaches. We next, experiment with different kinds of input and output ports. In the A|RT design flow, the designer has an option of choosing addressable or non-addressable ports. The former reads data from and writes data to a RAM (assuming all the input data is readily available), while the latter communicates with the external world through handshake signals [9]. The input and output ports used for obtaining the cycle count figures in Fig.4 referred to the addressable variety.

Table.3 shows the cycle count and register file sizes for input and output ports of the non-addressable variety. From the table it is clear that non-addressable inports and outports are preferable. The regfile size and cycle counts are larger when using addressable ports because of the overhead in

Table 3. *Addressable and Non-addressable ports comparison*

Type of port	Regfile(Kbits)	Cycle count
Addressable	3.454	41
Non-addressable	2.872	38

generating the address values for data to be read from the external RAM.

Table 4. *Scheduling algorithms comparison*

Type	Regfile(Kbits)	Cycle count
All	2.872	38
Asap	3.688	43
Alap	2.792	38
Alap Greedy	2.776	38

Table.4 depicts the various scheduling algorithms and their effects on regfile sizes and cycle counts. Depending on the scheduling algorithm chosen, the register life time can vary, and the size of the register file can change. For instance, the table clearly shows that the ASAP scheduling algorithm increases the register life time in our application. We choose the ALAP Greedy scheduling algorithm since it has the best cycle count, regfile file size combination. The final block diagram of the processor is as shown in Fig.5.

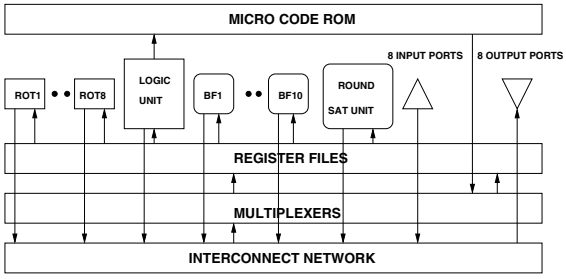

Figure 5. *Final block diagram*

3.5 Pipelining and interfacing to external hardware

The rotator is internally pipelined as explained in section 3.2. The DCT/IDCT processor has a four stage pipeline as shown in the left half of the Fig.6 where each stage refers to a stage in the flow graph in Fig.1 and has a latency of two. The next block of data can be read in only after the latency as shown in Fig.7. This is because a common data path has been used for computing the row wise and column wise transform. The row-wise transform for the next block

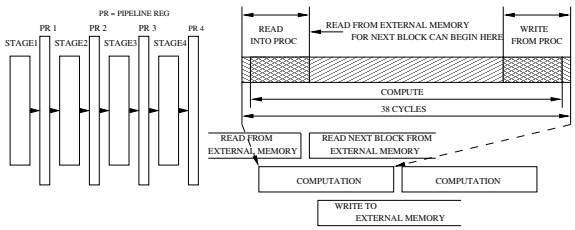

Figure 6. *Pipelining of Read-Compute-Write phases.*

cannot begin until the column-wise transform for the previous block is completed. For example, the data introduction interval is 26 cycles for an 8x2 subsampled block and 38 cycles for a 8x8 block (Fig.7).

The processor communicates with the external world through handshake signals [9]. All the cycle counts mentioned so far indicate the time taken for computing the forward or inverse DCT assuming the input data is available at the input port as soon as it is needed and the output data can be drained from the output port as soon as it becomes available. This is far from true in a realistic situation. If the input data is not available the processor will stall until it becomes available [9]. Similarly, if the data at the output ports is not drained immediately, the processor will stall again [9]. To reduce the frequency of stalls, the input and output data can be buffered external to the processor. The depth of the FIFO buffers will depend on the external read and write latency. With the introduction of such buffers, the read from external memory, computation and write to external memory can work in a pipeline as shown in the right half of Fig.6.

4 Results

The VLIW FDCT/IDCT processor shows good scaling in its performance. This is illustrated in Fig. 7. For example, the processor works in 24 cycles for a subsampled block of size 4x4, in 18 cycles for 2x4 etc.

4.1 Comparison with previous architectures

Table.5 compares the previous designs after scaling them to the technology used in the current design. The table clearly shows that only the proposed design fully supports scaled and sub-sampled blocks. The delay figures of the proposed design vary from 0.046 to 0.253μs. Thus for subsampled and scaled blocks, the proposed design is faster than all the existing architectures. The design proposed in [6] can be scaled in the sense that it can be adapted for higher throughput by adding data path units in parallel with a consequent increase in area. But with a fixed architecture, it is not scalable. The architecture proposed in [7] is par-

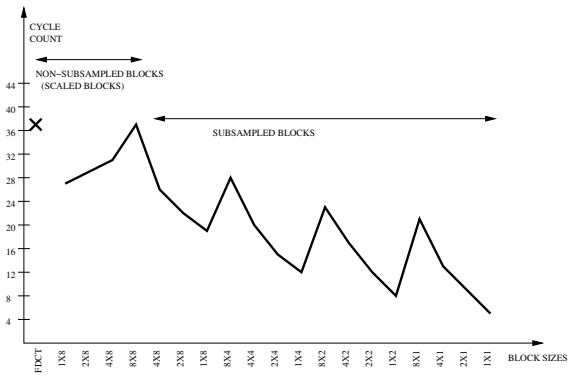

Figure 7. *Scaling performance of processor.*

tially scalable and is very compact. It can operate on 8x8 blocks or 2x4x8 blocks. Their proposed architecture is optimised for the digital VCR domain and does not meet the higher accuracy requirements of CCITT. (For e.g., the mean square error should be ≤ 0.083 for IDCT in the digital VCR domain, whereas the mean square error required by CCITT is ≤ 0.020.) If the accuracy requirements of CCITT need to be met by the architecture in [7], it would no longer be as compact. Hence, the proposed architecture is superior in its higher accuracy and support for sub-sampled and scaled blocks and offers significant advantages in comparison with the existing designs.

Table 5. *Comparison after scaling to 0.18μ CMOS technology*

Method	Latency(cycles)	Area(mm^2	Delay = latency/Freq.(μs)
[1]**	24	3.57	0.108
[2]*	128	1.0692	0.288
[3]*	135	0.588	0.2209
[5]*	72	5.4	0.216
[7]*	1208(fdct) 1192(idct)	0.072	10.98(fdct) 10.84(fdct)
[8]*+	54-108	0.196	0.1335 (4x8) 0.267(8x8)
Ours*++	7-38	0.834	0.046 - 0.253

** IDCT only * FDCT and IDCT

*+ FDCT and IDCT and partial support for scaled blocks

*++ FDCT and IDCT and full support for scaled and sub-sampled blocks

5 Conclusion

This paper showed the design of a fully scalable FDCT/IDCT VLIW processor. The core fully satisfies the accuracy requirements specified in CCITT. The processor can handle the high speed requirements of dual stream HD decoding. The presence of input data dependant control results in an excellent scaling performance of the processor.

Such multi-functional hardware ensures that the processor can find extensive use in applications where both forward and inverse DCT need to be performed (Eg. where both encoders and decoders are present locally). The proposed design offers specific advantages among others, in MPEG-2 decoders where scaled blocks can arise, in HDTV decoders where sub-sampled blocks can arise since there is a need to handle the Picture in Picture(PIP) functionality, and also in HD to SD down conversion in set top boxes. By working at a reduced latency for scaled and sub-sampled blocks, the proposed processor core can speed up MPEG-2 decoding.

The paper also discusses how the effects of each design decision on area and cycle time were analysed to arrive at the final optimal architecture. The comparison with the existing architectures shows that the proposed design is superior and offers many advantages.

References

[1] P. A. Ruetz et al., "A 160 Mpixels/s IDCT processor for HDTV", *IEEE Micro*, pp. 28-32, 1991.

[2] Uramoto et al., "A 100 Mhz 2-D Discrete Cosine Transform Core Processor", *IEEE Journal of Solid State Circuits*, Vol. 27, No. 4, pp. 492-499, Apr 1992.

[3] W. Li et al., "A high speed 2-D DCT/IDCT processor", *IEEE Intl. Symposium On Circuits And Systems*, Vol. 1, pp. 192-195, 1991.

[4] V. Srinivasan et al., "VLSI Design of high speed time recursive 2D DCT/IDCT processor for video applications", *IEEE Transaction On Circuits And Systems For Video Tech.*, Vol. 6, No. 1, pp. 87-96, Feb, 1996.

[5] H. Lim et al., "A Serial Parallel architecture for 2D DCT and IDCT", *IEEE Transaction On Computer*, Vol. 49, No. 12, pp. 1297-1309, Dec, 2000.

[6] T. Chang et al., "A simple processor core design for DCT/IDCT", *IEEE Transaction On Circuits And Systems For Video Tech.*, Vol. 10, No. 3, pp. 439-447, Apr, 2000.

[7] S. K. Paek et al., "A mode changeable 2D DCT/IDCT processor for digital VCR", *IEEE Transaction On Consumer Electronics*, Vol. 42, No. 3, pp. 606-616, Aug, 1996.

[8] C. Loeffler et al., "Practical Fast 1-D DCT Algorithms with 11 multiplications", *Intl. Conf. On Acoustics, Speech and Signal Processing, ICASSP*, pp. 988-991, May, 1989.

[9] ART Designer Reference manual, Frontier Design, v2.3 rev15, Apr, 2001. *www.adelantetechnologies.com*.

[10] CCITT SGXV Working Party XV/1 Specialists Group On Coding For Visual Telephony, Document 584, Nov., 1989.

[11] K. E. Wires et al. "Combined Unsigned and Two's Complement Squarers", *Proc. of the 33 Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, pp. 1215-1219, Oct, 1999.

[12] K. R. Rao and P. Yip "Discrete Cosine Transform : algorithms, advantages, applications". *Academic Press*, 1990.