The Performance Analysis of Fast DCT Algorithms on Parallel Cluster Architecture

Atri Sanyal and Swapan K Samaddar, *Member, IACSIT*

***Abstract*—Clustered processor, having 16 clusters with 4 PE's in each cluster and each PE having two ALU, which can perform simultaneously, is taken. The PE's operate in a SIMD manner. The cluster and PE can transfer data from one another [1]. 4 FDCT algorithms: - namely i>Vetterli ii> Arai's iii> Loeffler's iv> Chen's are taken into consideration. Our proposed 1D FDCT algorithm, parallel DCT is introduced. Examining all five algorithms using Timing Table it is concluded that percentage utilization is highest in Parallel DCT than other algorithms making it the most suitable to use in a parallel Cluster Architecture.**

***Index Terms*—Cluster architecture, FDCT algorithm, JPEG, ParallelDCT, Timing Table.**

I. Introduction

Standard for image compression in current use is the JPEG (Joint Picture Experts Group) and the energy compression technique used in this standard is known as Discrete Cosine Transform

The n rows of an N point DCT matrix T are defined by:

1> For all i=1 to n : ($t_{1i}=\sqrt{1/n}$)

2> For all i=1 to n and k=2 to n :

($t_{ki}=\sqrt{2/n}\cos((\prod(2i-1)(2k-1))/2n)$

The 8 point DCT matrix T (n=8) is:

0.3536 0.3536 0.3536 0.3536 0.3536 0.3536 0.3536 0.3536
0.4904 0.4157 0.2778 0.0975-0.0975-0.2778 –0.4157 -0.4904
0.4619 0.1913 -0.1913 -0.4619 -0.4619 -0.1913 0.1913 0.4619
0.4157 -0.0975 -0.4904 -0.2778 0.2778 0.4904 0.0975 -0.4157
0.3536 -0.3536 -0.3536 0.3536 0.3536 -0.3536 0.3536 0.3536
0.2778 -0.4904 0.0975 0.4157 -0.4157 -0.0975 0.4904 -0.2778
0.1913 -0.4619 0.4619 -0.1913 -0.1913 0.4619 -0.4619 0.1913
0.0975 -0.2778 0.4157 -0.4909 0.4904 -4157 0.2778 -0.0975

An n-point transform is defined as :

Y(1) x(1) t11…….t1n
….. = T x …… where T= . . Y(n) x(n)
. .
tn1.....tnn

The 8-point DCT is the basis of the JPEG standard, as well as several other standards such as MPEG-1 and MPEG-2. (for TV and video) and H.263 (for video-phones). Hence we shall concentrate on it as our example, but bear in mind that DCTs may be defined for a wide range of sizes n. The basic n-point DCT requires n^2 multiplications and n(n-1) additions to calculate y=T * X

For an 8x8 matrix that amounts to 64 multiplications and 56 additions.From DCT matrix, it is clear that symmetries exist in the DCT basis functions. These symmetries can be exploited to reduce the computation load of the DCT.

Manuscript received February 28, 2012; revised April 25, 2012.

A. Sanyal is with the NSHM College of Management and Technology, Kolkata, West Bengal, India (e-mail: atri.sanyal@nshm.com).

S. Samaddar is with School of Engineering, West Bengal University of Technology, Kolkata, and West Bengal, India (e-mail: swapansamaddar@gmail.com)

II. The Cluster Architecture and the Algorithm

Here we explored 4 of the known FDCT algorithm namely1>Vetterli's 2> Arai's 3>Loefflers 4>Chen'sAll of them are 1D FDCT algorithm except the third one which is a 2D FDCT algorithm.

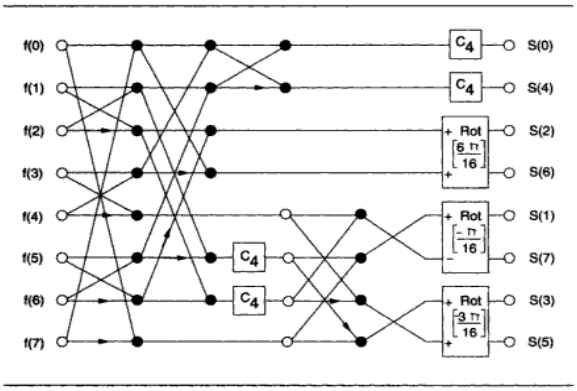


Fig. 1. Vetterli's dataflow diagram[1]

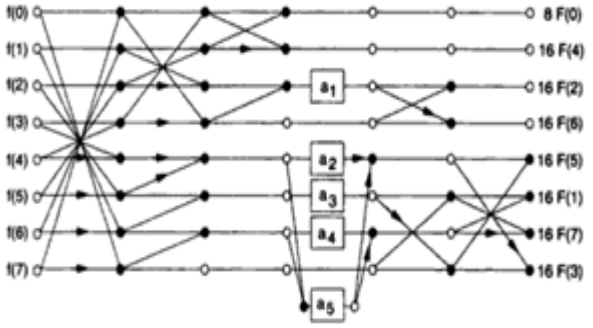


Fig. 2. Arai's dataflow diagram [2]

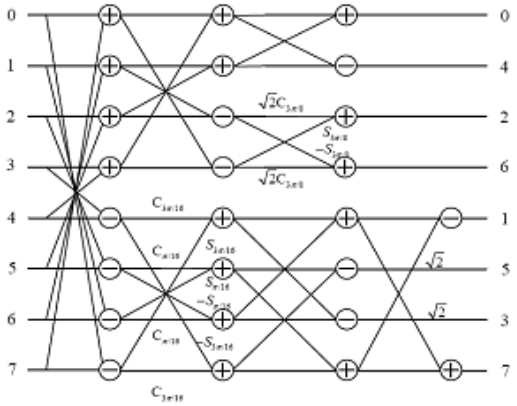


Fig. 3. Loeffler's dataflow diagram [3]

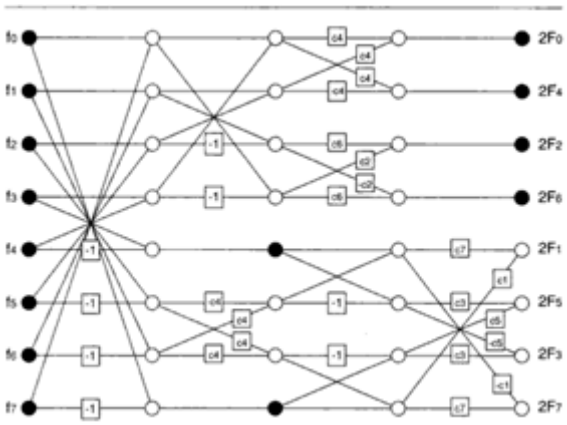


Fig. 4. Chen's dataflow diagram [4]

The clustered processor has 16 clusters with 4 processing elements (PE's) in each cluster. Each PE is having two ALU ohhkwhich can perform simultaneously. The PE's operate in a SIMD manner. The cluster and PE can transfer data from one another.

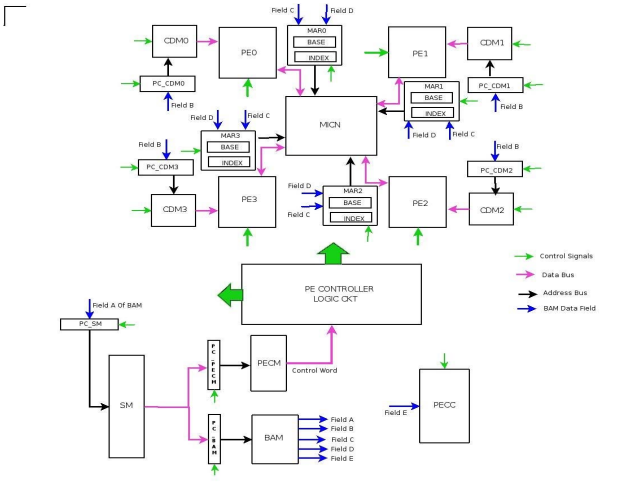


Fig. 5. Architecture of reconfigurable DSP processor of [5]

Each algorithm takes a row of an 8x8 matrix and computes the result using 2 clusters. So, all the 16 clusters can be used to compute the 8 rows DCT matrix multiplication paralley so that the entire computation can be done in that time.

III. Execution Methods of the Algorithm in the Cluster Processor

We will take Vetterli's algorithm as an example to understand how the algorithms can be mapped in the assumed architecture. The entire dataflow design of Vetterli's is decomposed vertically into 6 steps. For each step in the design the small nodes are the processing nodes. The first four processing nodes are mapped to the 4 PE's of cluster 0 and the next four processing nodes are mapped to the 4 PE's of cluster 1. The 6 steps of dataflow design become 6 stages of the cluster processor algorithm. In each step in the dataflow design, any processing node is getting its one operand from the same node and another from some other node in the design. For cluster processor algorithm one operand is simply the stored result of the previous stage. The second operand can be obtained by inter PE communication or inter cluster communication. If the second operand is coming from a PE which is not in the same cluster the first type of data communication is executed else the second one. As the second type of data communication is more time consuming, so we always try to arrange the mapping of processing node to PE to minimize the no of inter cluster communication. The data communication portion of the algorithm is shown below for Stage 5 of Cluster1 of the Vetterli's Algorithm.

```
Cl (1):
PAR
        Transfer DR0 of PE (0)←RES0 of PE(2)
        //intra cluster communication
        Transfer DR0 of PE (1)←RES0 of PE(3)
        //intra cluster communication
        Transfer DR0 of PE (2) ←RES0 of PE (0)
        //intra cluster communication
        Transfer DR0 of PE (3) ←RES0 of PE (1)
        //intra cluster communication
End PAR
```

After obtaining the operands using inter PE or inter cluster communication the instructions corresponding to the data lines are executed in the mapped PE's.

The stage 4 of cluster 1 executing addition operation of Vetterli's algorithm is shown below.

```
PAR
        For PE (j) where j=0,1
                ALU0←CR1+ DR0
                RES0←ALU0
        End For
End PAR
```

The subtract operation is exactly the same except for addition sign.

The stage 5 of cluster 0 executing multiplication operation of Vetterli's algorithm is shown below.

```
Load CR1 of PE (0) with C4
For PE (0)
        ALU0⇓CR1*DR0
        RES0⇓ALU0
End For
```

The last three boxes in the 6th stage is rotation box. The rotation of 2 input's x and y by an angle Q to produce a new representation X and Y is given by:

X=xCos (Q) + ySin (Q)

Y= -xSin (Q) + yCos (Q)

If angle of rotation is K∏/16 we get

X= Ckx + Sky

Y= -Skx + Cky

Where Ck and Sk are cos (k∏/16) and sin (k∏/16) respectively.

The rotation operation is mapped in the cluster processor using the following segment of algorithm.

```
For PE (i)
        ALU0←CR0*DR1 //alu0=x cos Q
        TR0←ALU0, ALU1← CR1*DR1
        //tr0= xcos Q, alu1=xsin Q
        TR1←ALU1,ALU0←CR0*DR0
        //tr1=xsinQ,alu0=ycosQ
        ALU1←CR1*DR0
        //alu1=ysin Q
        ALU0← ALU0- TR1
        //alu0=ycosQ-xsin Q
```

RES0←ALU0,ALU1←TR0+ALU1
//RES0=Y
RES1←ALU1 //RES1=X
End For

To further understand how this mapping procedure is done, we take two approaches:

A. The sixth processing node of the dataflow design is taken as PE (5) and the cluster processor algorithm through the entire stages is shown for that PE.

Stage 1 & 2:
ALU0←f (5) + f (6)
Stage 3:
Transfer: DR←DR of [Cl (0), PE (1)]
//inter cluster data comm.
ALU0←DR-CR
//CR stores the previous result
Stage 4:
ALU0←C4*CR
//CR stores the previous result
Stage 5:
Transfer: DR←DR of [Cl (1), PE (3)]
//inter PE data comm.
ALU0←DR+CR
//CR stores the previous result
Stage6:
//+Rot (∏/16): X=xCOS (∏/16) +ySIN (∏/16)
Transfer: DR←DR of [Cl (1), PE (0)]
//inter PE data comm.
ALU0←CR0*DR1
//alu0=x cos Q
TR0←ALU0, ALU1←CR1*DR1
//tr0= xcosQ, alu1=xsinQ
TR1←ALU1,ALU0←CR0*DR0
//tr1=xsinQ,alu0=ycosQ
ALU1←CR1*DR0 //alu1=ysinQ
ALU0←ALU0- TR1 //alu0=ycosQ-xsinQ
RES0←ALU0,ALU1←TR0+ALU1 //RES0=Y
RES1←ALU1 //RES1=X

B. The cluster-processor algorithm for a single stage i.e Stage 2 is shown for all PE's

PAR
Cl(0):
PAR
Transfer DR0 of PE(3)← RES0 of PE(0)
// intra cluster communication
Transfer DR0 of PE(0)← RES0 of PE(3)
// intra cluster communication
Transfer DR0 of PE(1)←RES0 of PE(1) of cluster (1)
//inter cluster communication
Transfer DR0 of PE(2)← RES0 of PE(2) of cluster (1)
// inter cluster communication
End PAR
PAR
For PE(j) where j= 0,1
ALU0←CR1+DR0 //arithmetic op
RES0←ALU0
End For
ENDPAR
For PE(2)
ALU0←CR1-DR0 //arithmetic op
ALU0←RES0
End For
For PE(3)
ALU0←DR0-CR1 //arithmetic op
ALU0←RES0
End For
Cl(1):
PAR
Transfer DR0 of PE(1)←RES0 of PE(1) of cluster (0)
//inter cluster communication
Transfer DR0 of PE(2)←RES0 of PE(2) of cluster (0)
// inter cluster communication
End PAR
For PE(1)
ALU0←CR1+DR0
RES0←ALU0
End For
For PE(2)
ALU0←DR0-CR1
RES0←ALU0
End For
End PAR

The Timing Table for the stage 2 will look like this:

TABLE I: TIMING TABLE FOR STAGE2 OF VETTERLI'S ALGORITHM

Cl0	PE0	S_{PDC}/ R_{PDC}	T_{add}	T_{dt}			
	PE1	S_{PDC}/ R_{PDC}	T_{add}	T_{dt}			
	PE2	S_{PDC}/ R_{PDC}	--	---	T_{add}	T_{dt}	
	PE3	S_{PDC}/ R_{PDC}	---	---	T_{add}	T_{dt}	
Cl1	PE0	----	---	---	---	---	--
	PE1	S_{PDC}/ R_{PDC}	T_{Load}	T_{Mul}, T_{Mul}	T_{dt}, T_{dt}	T_{add}	T_{dt}
	PE2	S_{PDC}/ R_{PDC}	T_{Load}	T_{Mul}, T_{Mul}	T_{dt}, T_{dt}	T_{add}	T_{dt}
	PE3	------	---	---	---	---	--
T.U	0	1	2	3	4	5	6

IV. OUR PROPOSED ALGORITHM-PARALLELDCT

It is assumed that the constants of the DCT matrix are represented as follows:

A=0.3536,B=0.4904,C=0.4157,D=0.2778,E=0.0975,F=0.4619,G=0.1913

Further it is assumed that the summation of the input matrix is represented as follows:

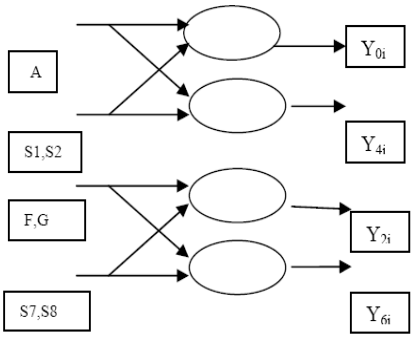
$S1=(X_{0i}+X_{1i}+X_{2i}+X_{3i}+X_{4i}+X_{5i}+X_{6i}+X_{7i})$
$S2=(X_{0i}-X_{1i}-X_{2i}+X_{3i}+X_{4i}-X_{5i}-X_{6i}+X_{7i})$
$S3=(X_{0i}-X_{7i})$
$S4=(X_{1i}-X_{6i})$
$S5=(X_{2i}-X_{5i})$
$S6=(X_{3i}-X_{4i})$
$S7=(X_{0i}-X_{3i}-X_{4i}+X_{7i})$
$S8=(X_{1i}-X_{2i}-X_{5i}+X_{6i})$

So replacing these variables the original 8x8 DCT equation produces:

i= 0 to 7
$Y_{0i}=S1*A$
$Y_{1i}=S3*B+S4*C+S5*D+S6*E$

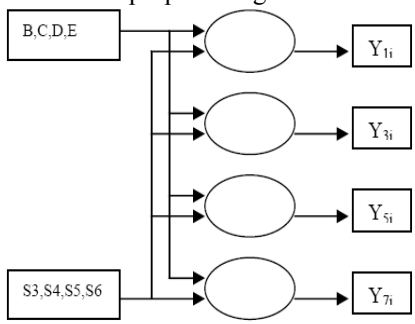
Y_{2i}=S7*F+S8*G
Y_{3i}= S3*C-S4*E-S5*B-S6*D
Y_{4i}=S2*A
Y_{5i}=S3*D-S4*B-S5*E+S6*C
Y_{6i}=S7*G-S8*F
Y_{7i}=S3*E-S4*D+S5*C-S6*B

We do not explore the symmetries further considering additions and subtractions. All the PEs have the local memory so adding a set of data from local memory is less complex than inter and intra cluster data communication and each of these take the same clock pulse (i.e. 1).[5]



Cluster0

The dataflow of the proposed algorithm is following:



Cluster1

Fig. 6. Data flow diagram for Cluster 0 & 1

Now the algorithm for the Cluster 1 is shown below:

```
Par
    For PE(i) where i=0 to 3
        Load CR0 and DR1 with Xi's;
    End For
End PAR
For PE(i) where i=0 to 3
    ALU0←CR0-DR1;
    i++;
End For
PAR
    For PE(i) where i=0 to 3
        LOAD CR1 with CONSTi's
        ALU0←CR0*DR1;
    End For
End PAR
PAR
    For PE(i) where i=0 to 3
        LOAD CR1 with CONSTi's
        ALU0←CR0*DR1;
    End For
End PAR
PAR
    For PE(i) where i=0 to 3
        LOAD CR1 with CONSTi's
        ALU0←CR0*DR1;
    End For
End PAR
PAR
    Transfer CR(0) of PE(0)←Mem[i] of PE(1);
    Transfer CR(0) of PE(1)←Mem[i] of PE(0);
    Transfer CR(0) of PE(2)←Mem[i] of PE(0);
    Transfer CR(0) of PE(3)←Mem[i] of PE(0);
End PAR
PAR
    Transfer CR(0) of PE(0)←Mem[i] of PE(2);
    Transfer CR(0) of PE(1)←Mem[i] of PE(2);
    Transfer CR(0) of PE(2)←Mem[i] of PE(1);
    Transfer CR(0) of PE(3)←Mem[i] of PE(1);
End PAR
PAR
    Transfer CR(0) of PE(0)←Mem[i] of PE(3);
    Transfer CR(0) of PE(1)←Mem[i] of PE(3);
    Transfer CR(0) of PE(2)←Mem[i] of PE(3);
    Transfer CR(0) of PE(3)←Mem[i] of PE(2);
End PAR
For PE(0)
    ALU0←CR0+DR1;
    i++;
End For
PAR
    For PE(i) where i=1 to 3
        ALU0←CR0-DR1;
        i++;
    End For
End PAR
PAR
    For PE(i) where i=0,3
        ALU0←CR0+DR1;
        i++;
    End For
End PAR
PAR
    For PE(i) where i=1,2
        ALU0←CR0-DR1;
        i++;
    End For
End PAR
PAR
    For PE(i) where i=0,2
        ALU0←CR0-DR1;
        i++;
    End For
End PAR
PAR
    For PE(i) where i=1,3
        ALU0←CR0-DR1;
        i++;
    End For
```

End PAR

The Timing Table for the entire execution phase of Cluster 1 is shown below:

TABLE II: TIMING TABLE FOR CLUSTER 1 OF PARALLEL DCT

CL1	1	2	3	4	5	6	7	8
PE0	T_{Load}	T_{Add}	T_{Load}	T_{Mul}	T_{Load}	T_{Mul}	T_{Load}	T_{Mul}
PE1	T_{Load}	T_{Add}	T_{Load}	T_{Mul}	T_{Load}	T_{Mul}	T_{Load}	T_{Mul}
PE2	T_{Load}	T_{Add}	T_{Load}	T_{Mul}	T_{Load}	T_{Mul}	T_{Load}	T_{Mul}
PE3	T_{Load}	T_{Add}	T_{Load}	T_{Mul}	T_{Load}	T_{Mul}	T_{Load}	T_{Mul}

CL1	9	10	11	12	13	14	15	16
PE0	T_{Load}	T_{Mul}	S_{PDC} /R_{PDC}	S_{PDC} /R_{PDC}	S_{PDC}/ R_{PDC}	T_{Add}		T_{Add}
PE1	T_{Load}	T_{Mul}	S_{PDC} /R_{PDC}	S_{PDC} /R_{PDC}	S_{PDC}/ R_{PDC}		T_{Add}	
PE2	T_{Load}	T_{Mul}	S_{PDC} /R_{PDC}	S_{PDC} /R_{PDC}	S_{PDC}/ R_{PDC}		T_{Add}	
PE3	T_{Load}	T_{Mul}	S_{PDC} /R_{PDC}	S_{PDC} /R_{PDC}	S_{PDC}/ R_{PDC}		T_{Add}	T_{Add}

CL1	17	18	19
PE0		T_{Add}	
PE1	T_{Add}		T_{Add}
PE2	T_{Add}	T_{Add}	
PE3			T_{Add}

Abbreviations:

$T_{add/sub}$=time taken for addition/subtraction,T_{DT}=time taken for register data transfer, S_{CDC}=time taken for Send instruction of Cluster data communication, R_{CDC}=time taken for Receive instruction of Cluster data communication, S_{PDC} =time taken for Send instruction of PE data communication, R_{PDC}=time taken for Receive instruction of PE data communication, T_{mul}=time taken for multiplication

T_{Load}=time taken for loading of data.

V. ANALYSIS AND CONCLUSION

For all five algorithms i.e. Vetterli's, Arai's, Loeffler, Chen's and our Parallel DCT we construct the algorithms from the instruction set taken from $_1$, Timing Table is constructed. Form the Timing Table we have found out the following details.

TABLE III: THE NO OF DATA COMMUNICATIONS (BOTH INTER CLUSTER AND INTER PE)

No of DC's	Vetterli's	Arai's	Loeffler's	Chen's	ParallelDCT
Inter PE Communication	7	20	18	18	11
Inter Cluster Communication	2	0	8	8	0

TABLE IV: THE NO OF CLOCK PULSE FOR EXECUTIONS OF INSTRUCTIONS IN EACH ALGORITHM (IT IS ASSUMED MULTIPLICATION TAKES 4 CLOCK PULSES AND ALL OTHER INSTRUCTIONS TAKE ONE CLOCK PULSE ACCORDING TO[1])

Vetterli's	Arai's	Loeffler's	Chen's	ParallelDCT
62	71	45	41	37

TABLE V: THE NO OF DIFFERENT INSTRUCTIONS FOR EACH ALGORITHM

	Vetterli's	Arai's	Loeffler's	Chen's	ParallelDCT
No of additions:	30	30	26	26	38
No of multiplications:	17	14	14	28	22
No of register data transfer:	32	47	46	54	---
No of load instructions:	15	14	8	26	30

TABLE VI: THE NO OF CLOCK PULSE FOR EXECUTIONS OF INSTRUCTIONS IN EACH ALGORITHM

	Vetterli's	Arai's	Loeffler's	Chen's	ParallelDCT
CP No of additions:	16	15	10	8	12
CP No of multiplications:	8	7	5	4	6
CP No of register data transfer:	17	23	15	12	----
CP No of load instructions:	7	7	3	10	8

TABLE VII: AVG % UTILIZATION OF THE CLUSTERS

Vetterli's	Arai's	Loeffler's	Chen's	Parallel DCT
34.21053	30.75	44.16667	53.44828	68.42105

From the findings it becomes clear that the new simple algorithm of computing DCT performed better than the other ones because of its high rate of parallelism inherent in its operation. The percentage utilization of the cluster processor is also significantly higher than the other algorithms. The other algorithms in the process of trying to minimize the no of multiplications have actually become lower in terms of % utilization of the processor.

REFERENCES

[1] M. Vetterli, "Fast 2-D discrete cosine transform," in *Proc. ICASSP*, Mar. pp. 1538–1541, 1985.

[2] Y. Arai, T. Aqui, M. Nakajima, "A fast DCT-SQ," *Scheme for images, Trans*, 1095-1097, 1988.

[3] C. Loeffler, A. Lightenberg, and G. Moschytz, "Practical fast 1-D DCT algorithms with 11multiplications," in *Proc. IEEE ICASSP*, vol. 2, pp. 988–991, 1989.

[4] W. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transfor," *IEEE, Trans. Commun. COMM-25*, pp. 1004-1009, 1977.

[5] P. Sinha, D. Basu, and A. Sinha, "A novel Architecture of a Reconfigurable Parallel DSP Processor," in *Proc. The 3rd Int. IEEE Northwest workshop on circuits and systems*, pp. 19-22, 2005.