

# Combined LNS Adder/Subtractors for DCT Hardware

Jie Ruan  
Department of  
Electrical and Computer Engineering  
Lehigh University  
Bethlehem, PA 18015  
Email: jir3@eecs.lehigh.edu

Mark G. Arnold  
Department of  
Computer Science and Engineering  
Lehigh University  
Bethlehem, PA 18015  
Email: marnold@eecs.lehigh.edu

## Abstract

*The Logarithmic Number System (LNS) shows good performance in low-precision computation, compared to traditional fixed-point and floating-point number systems. This paper exploits a way to share the table-lookup part of an LNS adder/subtractor pair for the Discrete Cosine Transform (DCT) in MPEG encoding. By this technique the equivalent computation in the DCT algorithm can be performed using much less area.*

## 1 Introduction

The Logarithmic Number System (LNS) has a larger dynamic range compared to the fixed-point number system. LNS is easier to implement at low precisions than the floating-point number system, which also has a similar dynamic range. The above features are suitable for MPEG encoding/decoding systems. In earlier work [1] and [2], LNS features were introduced into the Inverse Discrete Cosine Transform (IDCT), which is a key part of *Moving Picture Experts Group* (MPEG) video decoding and they made an extraordinary performance improvement. This paper analyzes Chen's fast DCT algorithm and exploits potential features to achieve less hardware. In Section 2, LNS is briefly introduced. Section 3 introduces the Berkeley encoding tool where the LNS features are introduced into MPEG encoding. Section 4 infers hardware implementation of Chen's fast DCT algorithm [3]. Section 5 shows how the LNS adders in the hardware of Chen's DCT algorithm can be combined by sharing the table-lookup part of two LNS adders, which achieves great area savings. Section 6 analyzes the area savings by using the combined LNS adder/subtractor method. Section 7 shows the areas of the actual synthesis results. Section 8 draws conclusions.

## 2 Logarithmic Number System (LNS)

The Logarithmic Number System (LNS) [4] represents a number by its exponent to a certain base  $b$ , such as  $b = 2$ . In LNS, one bit is used as the sign of the represented number and other bits are used for the exponent value.

The multiplication of two numbers is simply the sum of the two numbers' exponent parts. It only requires a fixed-point adder and an XOR gate, compared to the higher cost of a fixed-point multiplier.

However, the addition of two numbers is not a linear operation in LNS. The basic method for the addition calculation [2] is:

1.  $z = y - x$  and  $z_s = x_s \oplus y_s$ ;
2. If  $z_s = 0$ ,  $w = s_b(z)$ ; otherwise  $w = d_b(z)$ ;
3.  $t = x + w$ .

$x_s$  and  $y_s$  denote the sign bits of the two operands.

In Step 1, the calculation of  $z = y - x$  corresponds to calculating  $z = \log_b(Y/X)$ . Step 2 is a table lookup. Tables  $s_b(z)$  and  $d_b(z)$  correspond to addition or subtraction according to  $z_s$ :

- $s_b(z) = \log_b(1 + b^z)$ .
- $d_b(z) = \log_b |1 - b^z|$ .

Thus, the final result is  $t = \log_b |X(1 \pm Y/X)| = \log_b |X \pm Y|$ .

In this paper,  $W$  denotes the word size of an LNS number and  $F$  denotes the precision of the number's exponent, i.e., the fractional part of the exponent.

Figure 1 shows the diagram of an LNS adder. MUX 1 selects  $x - y$  or  $y - x$ , whichever is negative. Thus it avoids storing positive entries in the  $s_b(z)$  and  $d_b(z)$  tables. MUX 3 selects  $s_b(z)$  or  $d_b(z)$  depending on whether the control unit detects that  $x_s$  is the same as  $y_s$ . Because the value of  $s_b(z)$  is between 0 and 1 when  $z < 0$ , the  $s_b(z)$  table only needs to output  $F$  bits, instead of the wider word width needed for  $d_b(z)$ . In this way, the area of



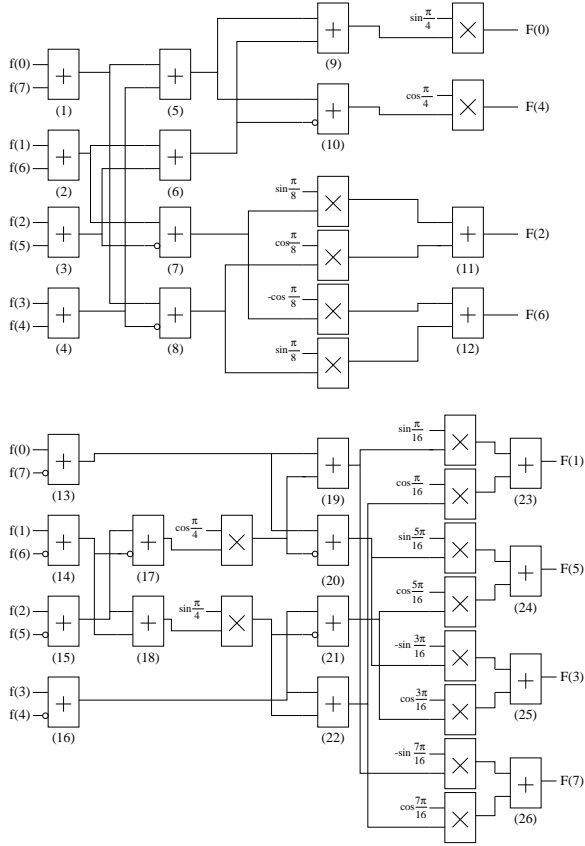


Figure 2: Hardware Diagram of Chen's DCT algorithm.

Figure 3 shows the diagram of a combined LNS adder/subtractor that could replace adders (9) and (10) in Figure 2. MUX 3 and MUX 4 select the proper results from the  $s_b(z)$  and the  $d_b(z)$  tables according to the signs of  $x$  and  $y$ . The two adders output the proper exponent parts of  $\log_2 |X + Y|$  and  $\log_2 |X - Y|$ . The two 1-bit MUXes (MUX 5 and MUX 6) output the proper sign bits. Altogether there are 10 adder pairs which can be optimized in this way: (1) and (13), (2) and (14), (3) and (15), (4) and (16), (5) and (8), (6) and (7), (9) and (10), (17) and (18), (19) and (20) and (21) and (22).

Further observing Figure 2, we can see that the upper inputs of adders (11) and (12) are both from the output of adder (7) and pass through the multipliers of  $\sin \frac{\pi}{8}$  and  $-\cos \frac{\pi}{8}$ ; thus, they are different in their signs. The lower input operands are both from the output of adder (8) and are multiplied by two numbers with same signs,  $\cos \frac{\pi}{8}$  and  $\sin \frac{\pi}{8}$ . This property can guarantee adders (11) and (12) do not access the same  $s_b(z)/d_b(z)$  table at the same time. So we can combine the two adders after the four multipliers and share the  $s_b(z)/d_b(z)$  tables of the two adders. Figure 4 shows the diagram of the functional unit combining adders (11) and (12). MUX 5 and MUX 6 select the proper result of  $|x_1 - y_1|$  or  $|x_2 - y_2|$  to send to the addresses of the  $s_b(z)$  and  $d_b(z)$  table. MUX 7 and

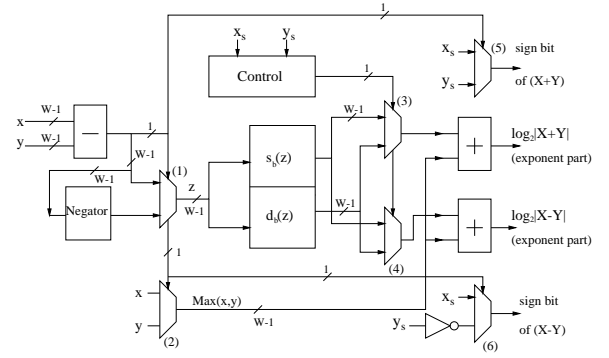


Figure 3: Combined LNS Adder/Subtractor with  $s_b(z)/d_b(z)$  Table Shared for  $X \pm Y$  and  $X \mp Y$ .

MUX 8 select the proper results from the  $s_b(z)$  and the  $d_b(z)$  table according to the sign of  $x_1$  and  $y_1$ . The two adders output the proper exponent parts of  $\log_2 |X_1 + Y_1|$  and  $\log_2 |X_2 + Y_2|$ . The two 1-bit MUXes (MUX 9 and MUX 10) output the proper sign bits. The circuit in Figure 4 is also substituted for adders (23) and (26) and adders (24) and (25).

The delay of Figure 3 is the same as the delay of Figure 1. Figure 4 has one more MUX in its critical path than Figure 1.

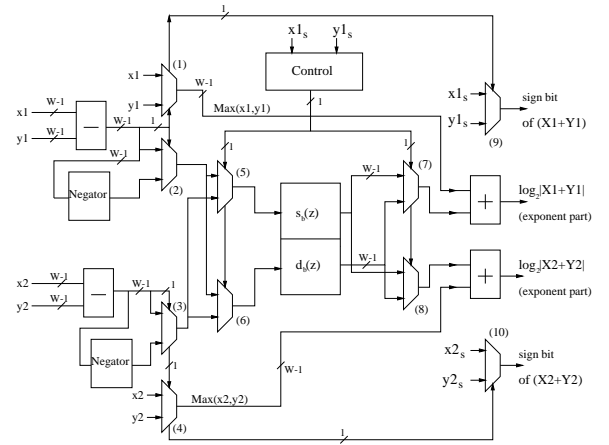


Figure 4: Combined LNS Adder/Subtractor with  $s_b(z)/d_b(z)$  Table Shared for  $X_1 \pm Y_1$  and  $X_2 \pm Y_2$ .

Figure 5 shows the diagram after the combination of the LNS adder/subtractors in Chen's algorithm, where all the 13 LNS adder pairs in Figure 2 are replaced by 10 combined LNS adder/subtractors from Figure 3 and 3 combined LNS adder/subtractors from Figure 4.

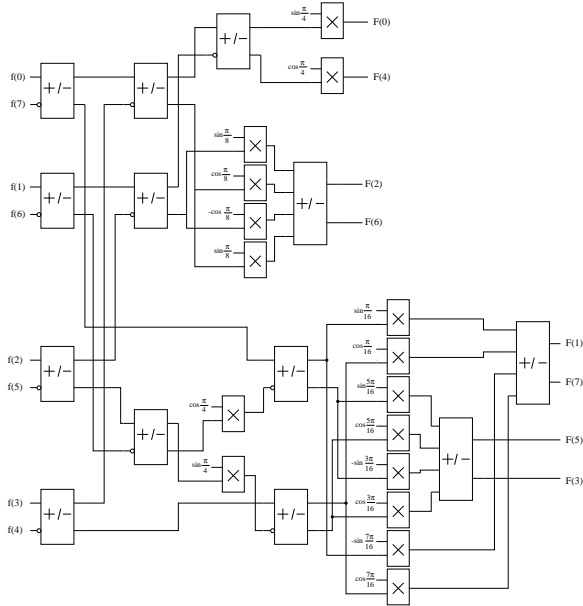


Figure 5: Hardware for Chen's algorithm after Substituting LNS Adder/Subtractors.

## 6 Area Analysis

The LNS adder in Figure 1 has one  $s_b(z)/d_b(z)$  table, 1 subtractor, 1 negator, 1 adder and 3  $(W - 1)$ -bit MUXes. The combined LNS adder/subtractor in Figure 3 has only one more  $(W - 1)$ -bit MUX and one more adder than the LNS adder. The combined LNS adder/subtractor in Figure 4 has one  $s_b(z)/d_b(z)$  table, 2 subtractors, 2 negators, 2 adders and 8  $(W - 1)$ -bit MUXes to realize the function of two LNS adders. Assuming the adders, the subtractors and the negators are all ripple-carry based, Table 1 shows the gate count of each component used in Figures 1, 3 and 4, excluding the control logic. Table 2

Component	INV	AND	OR	XOR	MUX
Adder	-	$3F + 13$	$2F + 8$	$2F + 9$	-
Negator	-	$F + 5$	-	$F + 5$	-
Subtractor	$F + 5$	$3F + 15$	$2F + 10$	$2F + 10$	-
MUX	-	-	-	-	$F + 5$

Table 1: The Gate Count of the Combinational Components.

shows the estimated combinational part gate count of the LNS adder in Figure 1, the novel LNS circuits in Figure 3 and Figure 4 and the LNS multiplier. Also, Table 2 shows how many synthesis "gates" each logic component needs, such as 3 gates per inverter. Considering this, the LNS circuits take  $101F + 491$ ,  $142F + 680$ ,  $214F + 1037$  and  $35F + 155$  gates respectively. The size of the combinational part of the LNS adder increases proportionally as the word length increases, while the size of the table-lookup part increases exponentially. The larger the precision, the larger percentage the table-lookup part

takes in the LNS adder, and the more the combined LNS adder/subtractor method saves in area.

In the hardware for Chen's DCT algorithm in Figure 2, all the 26 LNS adders can be substituted by 10 LNS adder/subtractors of Figure 3 and 3 LNS adder/subtractors of Figure 4. In this way, half the area of the table-lookup of the LNS adders can be saved. In the two-cycle hardware for Chen's DCT algorithm, 10 of 14 LNS adders can be substituted by 3 LNS adder/subtractors of Figure 3 and 2 LNS adder/subtractors of Figure 4. Thus, more than 1/3 of the table-lookup of the LNS adders can be saved.

Compon.	INV	AND	OR	XOR	MUX
Figure 1	$F + 5$	$7F + 33$	$4F + 18$	$5F + 25$	$3F + 16$
Figure 3	$F + 6$	$10F + 46$	$6F + 26$	$7F + 34$	$4F + 22$
Figure 4	$2F + 10$	$14F + 66$	$8F + 36$	$10F + 49$	$8F + 42$
Mul.	-	$3F + 13$	$2F + 8$	$2F + 10$	-
Gates	3	5	5	5	6

Table 2: The Combinational Gate Count for Different Precisions for LNS Adder and Combined LNS Adder/Subtractors.

Table 3 shows the area comparison of the combinational part (excluding table-lookup) between the one-cycle and two-cycle DCT hardware designs with/without the combined LNS adder/subtractors. At  $F = 4$ , 18% and 9% less area can be achieved by this combined adder/subtractor technique in one-cycle and two-cycle DCT hardware. The reason for a much lower percentage of area savings in two-cycle DCT hardware is because fewer LNS adders can be substituted by combined LNS adder/subtractors in the two-cycle DCT design. Another reason is that in the two-cycle DCT hardware, fewer adder pairs are substituted by Figure 3 and relatively more adder pairs are substituted by Figure 4. Figure 3 has only one extra MUX and one extra adder than Figure 1, while Figure 4 duplicates the combinational part of Figure 1.

One-cycle (without novel)	$3186F + 15246$
One-cycle (with novel)	$2622F + 12391$
Two-cycle (without novel)	$1866F + 9036$
Two-cycle (with novel)	$1710F + 8240$

Table 3: Area Comparison of Combinational Logic in DCT Hardware.

Figure 1, Figure 3 and Figure 4 have the same area for their table-lookup part,  $(2F + 5)2^{F+5}$  ROM bits. This area is not shown in Table 3.

## 7 Synthesis results

Mentor Graphics synthesis tool *Leonardo Spectrum* is used to obtain the area for the ordinary LNS adder in Figure 1 and the combined LNS adder/subtractors in Figure 3 and Figure 4. ASIC standard cell library *Sample SCL05U* is chosen as the target library. Table 4 shows

the synthesis result of the  $s_b(z)/d_b(z)$  table with different precisions. It shows that the actual synthesis result is much smaller than the theoretical estimated ROM bits. This is because the  $s_b(z)/d_b(z)$  table is sparse. Thus, the synthesis tool can significantly reduce area. At higher precisions, the ratio of theoretical estimation to actual synthesis result gets higher. This is because at larger word sizes, the more potential logic reduction can be exploited.

$F$	$W$	Theoretical	Synthesis	Ratio
		Est. (ROM bits)	Results (gates)	(Bits/Gate)
2	8	1152	188	6.13
3	9	2816	454	6.20
4	10	6656	873	7.62
5	11	15360	1839	8.35

Table 4: Area of Table-lookup Part with Different Precisions.

Table 5 lists the combinational area estimation and the actual synthesis results for Figure 1, Figure 3 and Figure 4. The synthesis results are smaller than estimated areas for the combinational part. The reason is that the synthesis tool uses more kinds of gates to reduce the area instead of just several basic kinds of gates in our theoretical estimation in Section 6.

$F$	$W$	Figure 1		Figure 3		Figure 4	
		Est.	Syn.	Est.	Syn.	Est.	Syn.
2	8	688	507	964	705	1465	1081
3	9	789	635	1106	861	1679	1308
4	10	890	647	1248	917	1893	1482
5	11	991	776	1390	1078	2107	1673

Table 5: Estimated Area and Synthesis Results for Combinational Part.

Table 6 shows the area comparison under different precisions using the above tool. The table shows that at a word length of 10, which is a suitable word length ( $W$ ) for LNS DCT implementation, the combined LNS adder/subtractor in Figure 3 has only 18% more area than an ordinary LNS adder but it can perform the same function as two LNS adders. For the combined LNS adder/subtractor in Figure 4, only 55% more area is needed to achieve the same function as two LNS adders.

$F$	$W$	Figure 1	Figure 3		Figure 4	
			Area	Ratio	Area	Ratio
2	8	695	893	1.28	1269	1.83
3	9	1089	1315	1.21	1762	1.62
4	10	1520	1790	1.18	2355	1.55
5	11	2615	2917	1.12	3512	1.34

Table 6: Area Comparison of LNS adder and Combined LNS Adder/Subtractors.

Table 7 lists the areas of one-cycle and two-cycle  $F = 4$  LNS DCT hardware. At  $F = 4$ , MPEG-1 encoding with LNS DCT can produce a visually acceptable result. Area of an equivalent fixed-point implementation is also listed. Table 8 shows the ratios of LNS DCTs to the equivalent one-cycle fixed-point DCT implementation with and

without the combined LNS adder/subtractor technique. It shows that LNS approaches require much less hardware than the fixed-point implementations for DCT. With the combined LNS adder/subtractor method, more hardware can be saved.

	Without	With	Fixed-point
One-cycle DCT	41505	28313	122443
Two-cycle DCT	23162	18059	77102

Table 7: Area Comparison of DCT hardware with/without combined LNS Adder/Subtractors ( $F = 4$ ).

	Without	With	Fixed-point
One-cycle DCT	0.34	0.23	1.00
Two-cycle DCT	0.19	0.15	0.63

Table 8: Area Ratio to the Fixed-point implementation with/without combined LNS Adder/Subtractors ( $F = 4$ ).

Many other fast DCT algorithms can also achieve area savings through the combined adder/subtractor approaches because of the butterfly units they use. For example, 24 of the 29 adders in [11] can be replaced by 12 combined LNS adder/subtractors shown Figure 3.

Power consumption is proportional to the area when other parameters in the system stay the same. So the area savings the LNS arithmetic achieved can greatly reduce the power consumption.

The conversion from LNS to fixed-point/floating-point representation also brings some overhead. The overhead of the conversion from LNS to fixed-point number with  $W = 10$  is 8 tables of  $512 \times 11$  bits and the overhead of the conversion from fixed-point to LNS is 8 tables of  $256 \times 9$  bits. Assuming the compression ratio (similar to Table 4) for the conversion tables is 7 bits/gate, the total conversion area overhead is about 35% of the LNS DCT. Such conversion hardware is used no more than half of the time. Thus its role in the total power consumption is reduced. Considering the input color signals can be sampled as LNS data at the beginning of MPEG encoding and the conversion of output DCT data to integers can be merged with variable length coding, the overhead is not always necessary to put on the LNS DCT hardware.

## 8 Conclusion

This paper introduces two combined LNS adder/subtractor designs to share the table-lookup part of LNS adders in a DCT algorithm by making use of properties of addition/subtractions in the DCT algorithm. Significant area savings can be achieved when this technique is used on the one-cycle and two-cycle DCT hardware for Chen's algorithm. This technique is suitable to reduce area and consequently power consumption in portable devices.

## References

- [1] Mark G. Arnold, "Reduced Power Consumption for MPEG Decoding with LNS," *Appl. Specific Arch. & Proc.*, pp. 65–75, July 2002.
- [2] Mark G. Arnold, "LNS for Low Power MPEG Decoding," *SPIE Adv. Signal Proc. Algorithms, Arch. & Implementations XII*, July 2002.
- [3] Wen-Hsiung Chen, C. Harrison Smith & S. C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. on Communication*, vol. COM-25, pp. 1004–1009, September 1977.
- [4] Israel Koren, *Computer Arithmetic Algorithms*. Prentice Hall, Inc, 1993.
- [5] John Watkinson, *The MPEG Handbook, MPEG-1, MPEG-2, MPEG-4*. Focal Press, 2001.
- [6] "Berkeley MPEG Tools," <http://bmr.c.berkeley.edu/research/mpeg>.
- [7] Jie Ruan & Mark G. Arnold, "LNS Arithmetic for MPEG Encoding Using a Fast DCT," *29th EUROMICRO CONFERENCE, Work-in-progress Session*, 2003.
- [8] John Makhoul, "A Fast Cosine Transform in One and Two Dimensions," *IEEE Trans. on Acoustics, Speech & Signal Proc.*, vol. ASSP-28, pp. 27–34, February 1980.
- [9] Jie Liang & Trac D. Tran, "Fast Multiplierless Approximations of the DCT With the Lifting Scheme," *IEEE Trans. on Signal Proc.*, vol. 49, pp. 3032–3044, December 2001.
- [10] Naoki Suehiro & Mitsutoshi Hatori, "Fast Algorithms for the DFT and Other Sinusoidal Transforms," *IEEE Trans. on Acoustics, Speech & Signal Proc.*, vol. ASSP-34, pp. 642–644, June 1986.
- [11] Byeong Gi Lee, "A New Algorithm to Compute the Discrete Cosine Transform," *IEEE Trans. on Acoustics, Speech & Signal Proc.*, vol. ASSP-32, pp. 1243–1245, December 1984.
- [12] Wen-Hsiung Chen & Harrison Smith, "Adaptive Coding of Monochrome and Color Images," *IEEE Trans. on Communications*, vol. COM-25, pp. 1285–1292, November 1977.
- [13] Avanindra Madiseti & Alan N. Willson, Jr., "A 100 MHz 2-D 8x8 DCT/IDCT Processor for HDTV Applications," *IEEE Trans. on Circuits & Systems for Video Tech.*, vol. 5, pp. 158–165, April 1995.
- [14] G. S. Taylor & G. M. Blair, "Design for the Discrete Cosine Transform in VLSI," *IEE Proc.-Comput. Digit. Tech.*, vol. 145, pp. 127–133, March 1998.
- [15] C. C. Chen & Y. Y. Chen, "Error Analysis of DCT Algorithms in Floating-point and Logarithmic Number Systems," *9th VLSI Design / CAD Symposium*, pp. 313–316, 1998.
- [16] Nam Ling & Nien-Tsu Wang, "A Real-Time Video Decoder for Digital HDTV," *Jnl. of VLSI Signal Processing*, vol. 33, pp. 295–306, 2003.
- [17] Jiun-In Guo & Jui-Cheng Yen, "An Efficient IDCT Processor Design for HDTV Applications," *Jnl. of VLSI Signal Proc.*, vol. 33, pp. 147–155, 2003.
- [18] Earl E. Swartzlander, Jr., D. V. Satish Chandra, H. Troy Nagle, Jr. & Scott A. Starks, "Sign/Logarithm Arithmetic for FFT Implementation," *IEEE Trans. on Comp.*, vol. C-32, pp. 526–534, June 1983.