RESOURCEFUL FAST DHT ALGORITHM FOR VLSI IMPLEMENTATION BY SPLIT RADIX ALGORITHM

M.Tamilselvi[1], M.P.Gomathi[2], A.Lakshminarayanan[3]

[1]*PG Scholar, Dept of ECE, Kongunadu College of Engg and Technology*
[2]*PG Scholar, Dept of ECE, Kongunadu College of Engg and Technology*
[3]*Asst.Professor, Dept of ECE, Kongunadu College of Engg and Technology*

Abstract

A new very large scale integration (VLSI) algorithmic rule for a 2^N-length discrete hartley transform (DHT) that may be expeditiously enforced on a extremely standard and parallel VLSI design having a regular structure is given. The DHT algorithmic rule may be expeditiously split on many parallel elements that may be dead at the same time. Moreover, the planned algorithmic rule is compatible for the subexpression sharing technique that may be used to significantly reduce the hardware complexness of the highly parallel VLSI implementation.

Keywords: *Discrete Hartley Transform (DHT), DHT Domain Processing, Fast Algorithms*

----------***----------

1 INTRODUCTION

THE discrete Fourier transform (DFT) is employed in several digital signal process applications as in signal and compression techniques, filter banks [1], signal representation, or analysis. The discrete hartley transform (DHT) [2], [3] will be used to with efficiency replace the DFT when the input sequence is real. There also are many split-radix algorithms for computing DHT with a low arithmetic price we've a extremely parallel resolution for the implementation of DHT supported a direct implementation of fast hartley transform (FHT). it's value to notice that hardware implementations of FHT are rare. Multipliers in an exceedingly VLSI structure consume a large portion of the chip space and introduce important delays.

To efficiently implement multipliers with lookup-table-based solutions, it's necessary that one quantity to be a constant. once one among the operands is constant, it's possible to store all the partial results in a read-only storage(ROM), and also the range of memory words is considerably reduced from 22L to 2L.The normal arithmetic operations are performed in several applications. In VLSI design additionally additions, multiplication like mathematical operation are performed, by using some transform operate the speed of operation can enlarged. The additions take less quantity of your time to execution, however multiplication take longer attributable to complicated operation. thus in this the rule development is focused to Multipliers. the simplest better-known transforms are those named for Laplace, Fourier, Hilbert, Hankel, Mellin, and Abelall of that still attract contributions to the mathematical literature.

2. SPLIT RADIX ALGORITHM

The split-radix FFT is a fast Fourier transform (FFT) algorithmic rule for computing the discrete Fourier transform(DFT), split radix could be a variant of the Cooley-Tukey FFT algorithmic rule that uses a mix of radices a pair of and 4: it recursively expresses a DFT of length N in terms of 1 smaller DFT of length N/2 and 2 smaller DFTs of length N/4 totally3 expression. The split-radix FFT, together with its variations, long had the excellence of achieving the lowest published arithmetic operation count (total precise variety of needed real additions and multiplications) to calculate a DFT of power-of-two sizes N. The split-radix algorithmic rule will solely be applied once N could be a multiple of four, however since it breaks a DFT into smaller DFTs it may be combined with the other FFT algorithmic rule as desired.

$$X_k = \sum_{n=0}^{N-1} x_n \left(\cos\frac{2\pi}{N}nk + \sin\frac{2\pi}{N}nk\right)$$

These are computed form, the decoputed form are

$$X_{2k} = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} (x_n + x_{n+(N/2)}) \times (\cos\frac{2\pi}{N/2}nk + \sin\frac{2\pi}{N/2}nk)$$

$$X_{4k+1} = \sum_{n=0}^{\left(\frac{N}{4}\right)-1} \left[\left(A_n + A_{\left(\frac{N}{4}\right)-n}\right)\cos\frac{2\pi}{N}n - \left(B_{n-}B_{\left(\frac{N}{4}\right)-n}\right)\sin\frac{2\pi}{N}n\right] \times (\cos\frac{2\pi}{N/4}nk + \sin\frac{2\pi}{N/4}nk)$$

$$X_{4k+3} = \sum_{n=0}^{\left(\frac{N}{4}\right)-1} \left[\left(A_n - A_{\left(\frac{N}{4}\right)-n}\right)\cos\frac{2\pi}{N}3n + \left(B_n + B_{\left(\frac{N}{4}\right)-n}\right)\sin\frac{2\pi}{N}3n\right] \times \left(\cos\frac{2\pi}{N/4}nk + \sin\frac{2\pi}{N/4}nk\right)$$

The operation of split radix in DHT is

$$N - \text{Point DHT} \rightarrow \frac{N}{2} - \text{Point DTH} + 2 \times \frac{N}{4} - \text{Point DHTs}$$

Split-radix refers to mixtures of 2 (or more) FFT engines, Split-radix FFTs have an identical structure to 2nd FFTs. Split-radix FFTs give bin spacing that produce higher results for several applications.

3. DHT TRANSFORM

The Hartley transform is an integral transform closely associated with the Fast Fourier Transform, however that transforms real-valued functions to real-valued functions. it absolutely was planned as an alternate to the Fourier transform by R. V. L. hartley in 1942, and is one in all several noted Fourier-related transforms. Compared to the Fourier transform, the hartley transform has the benefits of reworking real functions to real functions and of being its own inverse.
An FFT could be a thanks to work out identical result more quickly: computing the DFT of N points within the naive manner, victimization the definition, takes O(N2) arithmetic operations, whereas a FFT will work out an equivalent DFT in exactly O(N log N) operations. The distinction in speed is huge, particularly for long knowledge sets wherever N could also be within the thousands or millions.

FFTs ar of nice importance to a decent type of applications, from digital signal method and finding partial differential equations to algorithms for fast multiplication of enormous integers. within the presence of round-off error, several FFT algorithms are rather more correct than evaluating the DFT definition directly.

3.1 Forward and Inverse Transform

The forward and inverse discrete Hartley transform pair is given by

$$X_H(k) = \sum_{n=0}^{N-1} x(n)\,\text{cas}\frac{2\pi nk}{N}$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X_H(k)\,\text{cas}\frac{2\pi nk}{N}$$

Where casΦ = cos Φ + sin Φ.

$$H(u,v) = \frac{1}{MN} f(x,y).\left\{\cos(2\pi)\left(\frac{ux}{M} - \frac{vy}{N}\right) + \sin(2\pi)\left(\frac{ux}{M} - \frac{vy}{N}\right)\right\}$$

Where f(x ,y) is the intensity of the pixel at position(x,y) H(u,v)is the value of element in frequency domain. The results are periodic and the cosine+sine (CAS) term is call "the kernel of the transformation" (or "basis function").

The FHT are also used for fast arithmetic operation. In Fast Hartley Transform (FHT)Mand N must be power of 2. Which are much faster than DHT. These equation form representation is

$$H(u,v) = \{T(u,v) + T(M-u,v) + T(u,N-v) - T(M-u,N-v)\}/2$$

For both the FFT and the FHT the complexity is Nlog2N. An advantage of the DHT is the fact that the DHT is self-inverse, so that only one software routine or hardware device is needed for the forward and inverse FHT. For the forward and inverse FFT of a real signal, two different routines or devices are required.

The DHT is somehow conceptually easier than the DFT if the signaling is real, however all operations is applied with the FFT and also the FHT with an equivalent complexness.

3.2 Relationship to Fourier Transforms

This transform differs from the classic Fourier transform $F(\omega)=\mathcal{F}\{f(t)\}(\omega)$ in the choice of the kernel. In the Fourier transform, we have the exponential kernel: $\exp(-i\omega t) = \cos(\omega t) - i\sin(\omega t)$ where i is the imaginary unit.

The two transforms are closely related, however, and the Fourier transform (assuming it uses the same $\frac{1}{\sqrt{2\pi}}$normalization convention) can be computed from the Hartley transform via:

$$F(\omega) = \frac{H(\omega) + H(-\omega)}{2} - i\frac{H(\omega) - H(-\omega)}{2}$$

That is, the important and imagined elements of the Fourier transform are merely given by the even and odd elements of the Hartley transform, severally. Conversely, for real-valued functions f(t), the Hartley transform is given from the Fourier transform's real and imagined parts:

$$\{Hf\} = R\{\mathcal{F}f\} - I\{\mathcal{F}f\} = R\{\mathcal{F}f.(1+i)\}$$

Where R and that I denote the important and imagined elements of the complicated Fourier transform.

4 COMMON SUBEXPRESSION SHARING

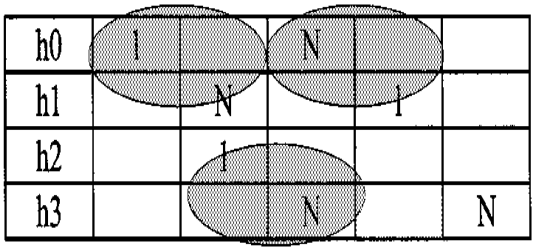

Fig 1 Coefficients and common sub expression

Where N denotes -1

Common subexpression sharing shares the subexpression among several multiplication-accumulation operations so that the total operation count is reduced. For example, Fig. 6 shows the FIR filter coefficients represented by the canonical signed digit (CSD) form. The circled teams of digits have an equivalent subexpression, in order that they will share an equivalent computation unit.

This approach is extremely effective for reducing the hardware value of multiple constant multiplications, particularly for the filter-like operation. Thus, by sharing the common subexpression, the number of additions is reduced (38% reduction).

The common subexpression part is first calculated, and then we shift or negate the subexpression for other computations. The hardware to generate different constant multiplications is called adder network in the paper. By using subexpression sharing, much computation can be saved if we can maximally find these common sub expressions.

5. LOOK-UP-TABLE

To efficiently implement multipliers with lookup-table-based solutions, it's necessary that one quantity to be a continuing. once one in all the operands is constant, it's potential to store all the partial leads to a read-only memory, and therefore the variety of memory words is considerably reduced from 22Lto 2L.

LUT in the main depends on RAM blocks. we tend to might understand any logic operate from truth table victimisation RAM, this can be done by mapping the inputs to the address bus and therefore the output is mapped to the information bus.

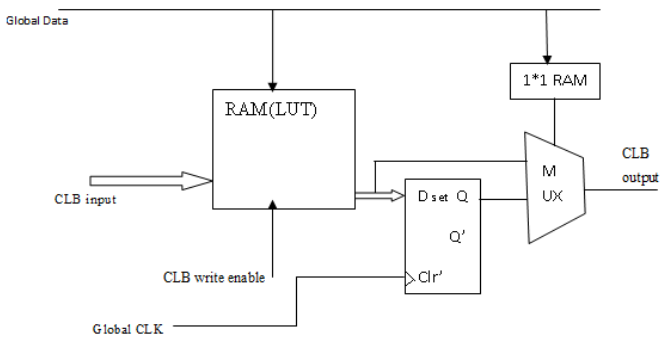

Fig 2 Block diagram of LUT

These Look-Up-Tables are very helpful to implement the multiplier factor in Field Programmable Gate Array (FPGA). The FPGA is one among the components in VLSI.

5.1 Parallel Process

Parallel version in the main used when combining a bigger FFT with a three or five purpose FFT, since it's possible to use three or five massive FFTs during a single device.

6. PROPOSED SYSTEM

Highly parallel and standard resolution for the implementation of type-III DHT supported a replacement VLSI rule is given. A extremely parallel resolution for the implementation of DHT supported a direct implementation of fast hartley transforms (FHT). it's value to notice that hardware implementations of FHT are rare.

A new very large scale integration (VLSI) algorithmic rule for a 2N-length discrete Hartley transform (DHT) which will be expeditiously enforced on a extremely standard and parallel VLSI design having an everyday structure is conferred. The DHT algorithmic rule will be expeditiously split on many parallel elements which will be dead at the same time. Moreover, the projected algorithmic rule is compatible for the subexpression sharing technique which will be wont to considerably reduce the hardware quality of the extremely parallel VLSI implementation. using the benefits of the projected algorithmic rule and also the fact that we will efficiently share the multipliers with constant, the quantity of the multipliers has been considerably reduced specified the quantity of multipliers is incredibly tiny examination with that of the existing algorithms. Moreover, the multipliers with a relentless will be expeditiously enforced in VLSI.

6.1 Parallel VLSI Architecture

In order to obviously illustrate the options and advantages of the planned formula, the VLSI design for a DHT of length N = 32 is given in Fig. 1(a) and (b). It may be seen that the planned design is very parallel and has a standard and regular structure

being formed of only a number of blocks: U, MUL, ADD/SUB, XCH, and a number of further adders/subtracters.

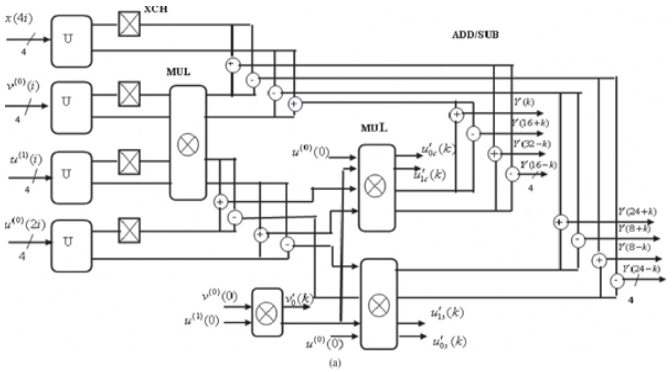

Fig 3 Block diagram

The "U" blocks implement (20), XCH blocks interchange the values and area unit merely enforced in hardware by applicable wiring, and MUL blocks are wont to implement the shared multipliers with a continuing. This block contains four multipliers with a constant. every multiplier factor is shared by four input sequences that are increased with an equivalent constant in associate interleaved manner using multiplexers and demultiplexers controlled by 2 clocks.

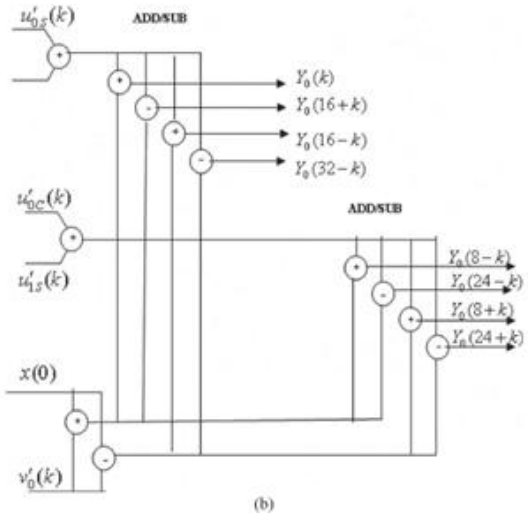

Fig 4 Addition/Subtraction

One of the benefits of this algorithmic rule and design is that the fact that the multiplications with an equivalent constant are shared within the MUL blocks. Thus, the amount of multipliers is considerably but the worth forty given in Table I that has become currently only sixteen. the final values Y(k) of Section A and Y0(k) of Section B are finally additional to get the output sequence Y (k)using a further adder not given in Fig. one for simplicity. The projected design includes a high throughput of thirty two samples per clock and might be pipelined. it's extremely parallel employing a low hardware complexness structure. multiprocessing is one in all the most important ways that to scale back power consumption, the high process speed being listed off for low power victimization the reduction of the availability voltage price. the desired management structure is extremely straightforward this can be another necessary advantage.

7. RESULTS

VHDL is used because the hardware description language due to the pliability to exchange among environments. The code is pure VHDL that would simply be implemented on alternative devices, while not dynamic the design. The Xilinx and ModelSim are used for implementation.

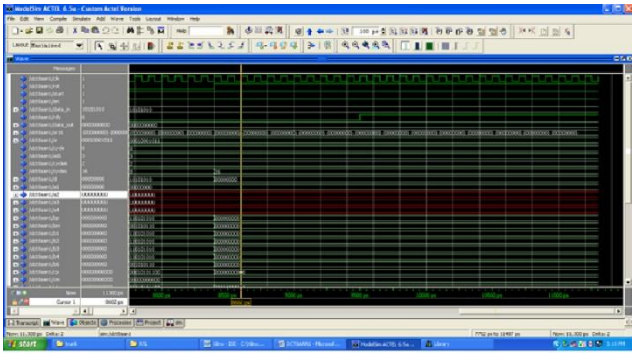

Fig 5 Input

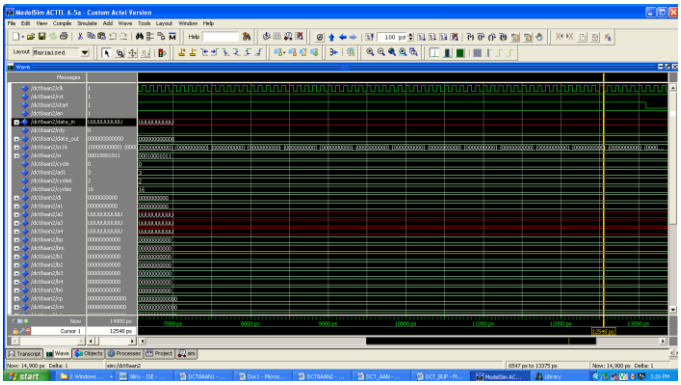

Fig 6 Output

8. CONCLUSIONS

The numbers of discrete transforms are obtainable for VLSI implementation. they need some difficult like that value, area, power, fastness and time. In projected work DHT transform is enforced. The DHT have a lot of advantage than different transforms.

The discrete Hartley transform (DHT) avoids complex arithmetic and it's own inverse. It needs [*fr1] the memory storage and N log2 N real operation instead advanced

operations of DFT and fewer operations might facilitate in truncation and miscalculation errors. These are the benefits of DHT transform.

REFERENCES

[1] D.F.Chiper, "A Novel VLSI DHT Algorithm for a Highly Modular and ParallelArchitecture," IEEE Trans.CircuitsSyst.II, Exp.Briefs,vol.60,no.5, pp.282-286,May 2013.

[2] Bracewell.R.N, "Discrete Hartley transform," J. Opt. Soc. Amer., vol.73, pp. 1832–1835, Dec. 1983.

[3] Bi.G, Chen. Y, and Zeng. Y, "Fast algorithms for generalized discrete Hartley transform of composite sequence lengths," IEEE. Trans. Circuits Syst. II, vol. 47,no. 9, pp.893-901, 2000.

[4] Bracewell.R.N. "Aspects of the hartley transform". Proc.IEEE, 82:381–387, March 1994.

[5] Chiper. D.F, Swamy. M.N.S, M. O. Ahmad, and Stouraitis .T, "A systolic array architecture for the discrete sine transform," IEEE Trans. SignalProcess., vol. 50, no. 9, pp. 2347–2354, Sep. 2002.

[6] Hu.N.C, Chang.H.I, and Ersoy.O.K, "Generalized discrete Hartley transforms," IEEE Trans. Signal Process.,vol. 40, no. 12, pp. 2931-2940, 1992.

[7] Malvar, "Fast Computation of the Discrete Cosine Transform and the Discrete Hartley Transform", IEEE Trans. on ASSP.,Vol.ASSP-35, N0.10, PP.1484-1485, OCt, 1987.

[8] Wang.Z, "A prime factor fast W transform algorithm," IEEE Trans.Signal Processing, vol. 40, pp. 2361–2368, Sept. 1992.

[9] Widrowetal.B, "Fundamental relations between the LMS algorithm and the DFT," IEEE Trans. Circuits Syst., vol. CAS-34, pp. 814–820,July 1987.

[10] Yang.R.Y and Lee.C.Y, "High-throughput data compressor designs using content addressable memory," in Proc. IEEE Int. Symp. CircuitsSyst. (ISCAS), London, May 1994, pp. 147–150.

[11] Z. Wang, "Comments on 'Generalized discrete Hartley transform',"IEEE Trans. Signal Processing, vol. 43, pp. 1711–1712, July 1995.