# Hardware-Efficient DFT Designs with Cyclic Convolution and Subexpression Sharing

Tian-Sheuan Chang, *Member, IEEE*, Jiun-In Guo, and Chein-Wei Jen, *Member, IEEE*

***Abstract*—This paper presents a hardware efficient design for the discrete Fourier transform (DFT). The proposed design not only applies the constant property, but also exploits the numerical property of the transform coefficients. DFT is first formulated as cyclic convolution form to make each DFT output sample computations have the same computation kernels. Then, by exploring the symmetries of DFT coefficients, the word-level hardware sharing can be applied, in which two times the throughput is obtained. Finally, bit-level common subexpression sharing can be efficiently applied to implement the complex constant multiplications by using only shift operations and additions. Though the three techniques have been proposed separately for transform, this paper integrates the above techniques and obtains additive improvements. The I/O channels in our design are limited to the two extreme ends of the architecture that results in low I/O bandwidth. Compared with the previous memory-based design, the presented approach can save 80% of gate area with two-times faster throughput for length $N = 61$. The presented approach can also be applied to power-of-two length DFT. Similar efficient designs can be obtained for other transforms like DCT by applying the proposed approach.**

***Index Terms*—Common subexpression sharing, cyclic convolution, DFT.**

## I. Introduction

THE DISCRETE Fourier transform (DFT) is an important function widely used in many image and signal processing applications. Because of the computational complexity, the fast algorithms which reduce the number of operations greatly or the hardware solutions to accelerate the computation can be adopted for many real-time applications. The fast Fourier transform (FFT) algorithms are attractive for low latency and high throughput, but its global communication complexity makes it unsuitable for VLSI hardware design. Although the DFT systolic array [1]–[7] is VLSI oriented because of its modularity and regularity; however, it is still severely limited by the area consumption. In the existing DFT systolic arrays, multipliers are the fundamental computing elements in PE's. Since multipliers consume a large silicon area, the limited chip size puts a severe limitation to the allowable number of PE's. To enable constant multiplications, the memory-based designs [8] proposed cyclic convolution-based architecture. The cyclic convolution-based design has the features of simple I/O behavior and removing data redundancy in the DFT coefficients. However, their designs are still not efficient enough, since they only apply the constant property of transform coefficients.

Manuscript received June 1999; revised May 2000. This paper was recommended by Associate Editory M. Bayoumi.

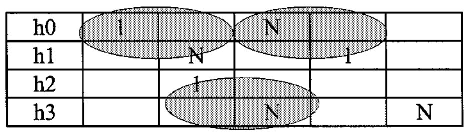
T.-S. Chang and C.-W. Jen are with the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

J.-I. Guo is with the Department of Electronics Engineering, National Lien-Ho College of Technology, Miao-Li, Taiwan, R.O.C.

Publisher Item Identifier S 1057-7130(00)07760-0.

This paper is to develop a hardware-efficient DFT architecture [9]. Since DFT coefficients are constant for fixed $N$-point transforms, we can expand constant multiplications with shift operations and additions and sharing the common ones. This technique is called common subexpression sharing [10]–[13]. Common subexpression sharing has been shown to be very effective to reduce the hardware cost of multiple constant multiplications, especially for filter-like operations. However, direct application of common subexpression sharing to the transforms is not efficient [12], since only limited bit-level sharing is considered. The proposed approach improves the sharing efficiency by first considering the whole word-level coefficient sharing and then the bit-level subexpression sharing. Such multi-level sharing not only reduces hardware cost greatly, but also increases the throughput by two times.

To facilitate multi-level sharing, we reformulate the DFT coefficient matrix as cyclic convolution form. Since coefficients in cyclic form are regularly propagated at the rows of the coefficient matrix, coefficient multiplication will have the maximum similarity between different output computations. So, we can use constant coefficient multiplications instead of variable multiplications to implement the DFT. After cyclic reformulation, we can further use word-level sharing to explore the numerical relationship of coefficients, e.g., the symmetrical property of DFT coefficients that is often used in FFT derivations. In this paper, we first use the even and odd symmetries of the cosine and the sine functions to combine multiplications with the same value of the sine or cosine constant together. This reduces the constant multiplication number by a factor of two. Then we take advantage of the symmetries of the sine and cosine as functions of frequency index. Thus, only half of the outputs need to be calculated. So we calculate two output values at a time and obtain two times the throughput. After applying word-level sharing to DFT, we can apply bit-level sharing to explore similarities of bit-level operations. Though the three design techniques of cyclic convolution formulation [8], [14], word-level sharing [15], and bit-level sharing [10]–[13] have been proposed previously and applied separately to the transform design, they achieve maximum hardware saving and throughput increasing when they are applied together to the designs. The first techniques reformulate the transform equations into cyclic convolution form to make each DFT output have the same computation kernels instead of multiple ones. Then the second and third techniques complement each other by exploring sharing both in word level and bit level. Comparing the design only with



| | | | | | |
|---|---|---|---|---|---|
| h0 | 1 | | N | | |
| h1 | | N | | 1 | |
| h2 | | 1 | | | |
| h3 | | | N | | N |

Fig. 1. CSD-represented coefficients and common subexpression, where $N$ denotes $-1$.

common subexpression sharing, the combined techniques can save half of the hardware cost at the same throughput.

This paper is organized as follows. We first introduce the technique of common subexpression sharing in Section II. Then, we present the algorithm formulation of DFT in Section III. Section IV illustrates the proposed architecture design based on Section III. Section V lists the performance and the comparisons of hardware cost with other designs. Finally, this paper is concluded in Section VI.

## II. Review of Common Subexpression Sharing

Common subexpression sharing shares the subexpression among several multiplication-accumulation operations so that the total operation count is reduced. For example, Fig. 1 shows the FIR filter coefficients represented by the canonical signed digit (CSD) form. The circled groups of digits have the same subexpression, so they can share the same computation unit. The filtering operation is

$$\begin{aligned} y =\, & x[0] - x[0] \gg 2 - x[1] \gg 1 + x[1] \gg 3 + x[2] \\ & \gg 1 - x[3] \gg 2 - x[3] \gg 4 \end{aligned} \tag{1}$$

where $x[a] \gg b$ denotes "$a$" sample delay of $x$ and then "$b$" digit right shifts of the signal. If we define common subexpression

$$w[i] = x[i] - x[i+1] \gg 1 \tag{2}$$

we can rewrite the filtering operation as

$$y = w[0] - w[0] \gg 2 + w[2] \gg 1 - x[3] \gg 4. \tag{3}$$

Thus, by sharing the common subexpression, the number of additions is reduced from six to four. Fig. 2 shows the computation flow of the filter example. 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 subexpressions.

The drawback of the existing subexpression sharing approaches is that they only deal directly with filter type operations as shown in Fig. 1. When they are applied to transforms such as DFT, they must be subsequently applied twice [12], which is less effective than optimizations that consider the entire solution space at once. Besides, unlike the filter input having delay dependency, the intermediate result, such as $w$ in the example, cannot be delayed and used by other summations since each input of the transform does not have any delay relationship. These problems severely limit the effectiveness

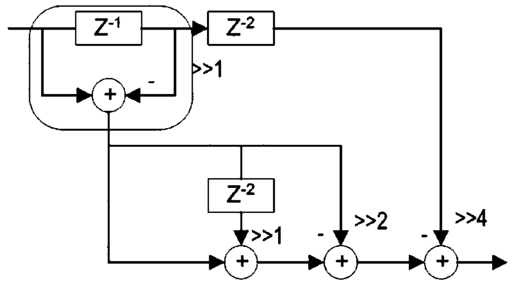


Fig. 2. Computation flow of the filter example.

of the existing subexpression sharing when it is applied to transform designs.

A promising approach to solve above problems is to reformulate transforms into filter type operations such that subexpression sharing methods can be applied. From the mathematical viewpoints, both transforms and filters use the inner products as their basic operations. On the other hand, from the viewpoint of the signal flow graph, input data of filter operations has the unit delay dependency, while input data of transforms does not. To create such pseudo-dependency, the transforms can be rewritten as filter operations. Therefore, this paper uses the algorithm based on the data permutations introduced in [8] (that are based on [14]) to reformulate transforms into filter operations with cyclic convolution. By using cyclic convolution, input data of the transforms are forced to have pseudo delay dependency and to be filter-like operations.

## III. Algorithm Formulation of DFT

### *A. Cyclic Convolution Formulation for DFT*

The 1-D DFT of the input sequence $\{x(i), i = 0, 1, \cdots, N-1\}$ can be generally expressed as

$$Y(k) = \sum_{n=0}^{N-1} y(n) \cdot W_N^{nk}, \qquad k = 0, 1, \cdots, N-1 \tag{4}$$

where $W_N^{nk}$ is assumed to be $e^{(-2nk\pi j/N)}$. From the derivation of our previous approaches [8] for prime length DFT, we can formulate (4) as

$$Y(0) = \sum_{i=0}^{N-1} x(i) \tag{5a}$$

and

$$Y(g^k) = T(g^k) + x(0), \qquad k = 1, \cdots, N-1 \tag{5b}$$

where

$$T(g^k) = \sum_{i=1}^{N-1} x(i) \times \exp\left(-j\frac{2\pi}{N} \times g^{i+k}\right), \quad k = 1, \cdots, N-1 \tag{5c}$$

and "$g^i$" denotes the result of "$g^i$ modulo $N$ operation" for short and "$g$" is a primitive element. From (5c), we can find that the sequence $T(k)$ is the cyclic convolution of the sequence

$\{x(i), i = 1, \cdots, N-1\}$ and the kernels $\{\exp(-(2j\pi/N) \times g^{i+k}), i$ and $k = 1, \cdots, N-1\}$.

Let us take a 1-D five-point DFT as an example. The 1-D five-point DFT of the input sequence $\{x(n),\ n = 0, 1, 2, 3, 4\}$ is expressed as

$$\begin{bmatrix} Y(0) \\ Y(1) \\ Y(2) \\ Y(3) \\ Y(4) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 & W^4 \\ 1 & W^2 & W^4 & W^6 & W^8 \\ 1 & W^3 & W^6 & W^9 & W^{12} \\ 1 & W^4 & W^8 & W^{12} & W^{16} \end{bmatrix} \times \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \end{bmatrix} \tag{6a}$$

where $W = \exp(-j2\pi/5)$. By using the formulations (5a)–(5c) and $g = 2$, a cyclic convolution form can be expressed as

$$\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(3) \\ Y(1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W^4 & W^3 & W^1 & W^2 \\ 1 & W^3 & W^1 & W^2 & W^4 \\ 1 & W^1 & W^2 & W^4 & W^3 \\ 1 & W^2 & W^4 & W^3 & W^1 \end{bmatrix} \times \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(3) \\ x(1) \end{bmatrix} \tag{6b}$$

or in an alternative form as

$$Y(0) = \sum_{n=0}^{4} x(n)$$
$$\begin{bmatrix} Y(2) \\ Y(4) \\ Y(3) \\ Y(1) \end{bmatrix} = \begin{bmatrix} x(1) & x(2) & x(4) & x(3) \\ x(3) & x(1) & x(2) & x(4) \\ x(4) & x(3) & x(1) & x(2) \\ x(2) & x(4) & x(3) & x(1) \end{bmatrix} \begin{bmatrix} W^2 \\ W^4 \\ W^3 \\ W^1 \end{bmatrix} + \begin{bmatrix} x(0) \\ x(0) \\ x(0) \\ x(0) \end{bmatrix}. \tag{6c}$$

This equation shows a commutative representation of the cyclic convolution in previous form. In (6c), each DFT output computation except $Y(0)$ shares the same kernel instead of multiple kernels as in (6a). By using above formulation, (6c) can be implemented by a constant coefficient filter-like architecture instead of a multiplier-based design, which can save much area cost.

## B. Symmetry Exploration of the Cyclic DFT

Take the 1-D five-point DFT as an example. First we expand the DFT coefficients and notice the even symmetries of cosine function $c_N^i = c_N^{N-i}$, $i = 1, \cdots, N-1$, where $c_N^i = \cos(2 \cdot i \cdot \pi/N)$, and the odd symmetries of sine function, $s_N^i = -s_N^{N-i}$, $i = 1, \cdots, N-1$, where $s_N^i = \sin(2 \cdot i \cdot \pi/N)$

$$\begin{bmatrix} W^2 \\ W^4 \\ W^3 \\ W^1 \end{bmatrix} = \begin{bmatrix} c_5^2 - js_5^2 \\ c_5^4 - js_5^4 \\ c_5^3 - js_5^3 \\ c_5^1 - js_5^1 \end{bmatrix} = \begin{bmatrix} c_5^2 - js_5^2 \\ c_5^4 - js_5^4 \\ c_5^2 + js_5^2 \\ c_5^4 + js_5^4 \end{bmatrix}. \tag{7}$$

So, we can combine the multiplications with the same constant cosine coefficient together. This reduces the constant multiplications by a factor of two. Exploiting the symmetry properties of cosine and since functions, the sequence $\{Y(i),\ i = 1, 2, 3, 4\}$ in (6c) can be expressed as

$$\begin{aligned}
\begin{bmatrix} Y(2) \\ Y(4) \\ Y(3) \\ Y(1) \end{bmatrix}
&= \begin{bmatrix} x(1) & x(2) & x(4) & x(3) \\ x(3) & x(1) & x(2) & x(4) \\ x(4) & x(3) & x(1) & x(2) \\ x(2) & x(4) & x(3) & x(1) \end{bmatrix} \begin{bmatrix} c_5^2 - js_5^2 \\ c_5^4 - js_5^4 \\ c_5^2 + js_5^2 \\ c_5^4 + js_5^4 \end{bmatrix} + \begin{bmatrix} x(0) \\ x(0) \\ x(0) \\ x(0) \end{bmatrix} \\
&= \left[\begin{array}{cc} x(1)+x(4) & x(2)+x(3) \\ x(3)+x(2) & x(1)+x(4) \\ \hline x(4)+x(1) & x(3)+x(2) \\ x(2)+x(3) & x(4)+x(1) \end{array}\right] \begin{bmatrix} c_5^2 \\ c_5^4 \end{bmatrix} \\
&\quad - j \left[\begin{array}{cc} x(1)-x(4) & x(2)-x(3) \\ x(3)-x(2) & x(1)-x(4) \\ \hline x(4)-x(1) & x(3)-x(2) \\ x(2)-x(3) & x(4)-x(1) \end{array}\right] \begin{bmatrix} s_5^2 \\ s_5^4 \end{bmatrix} + \begin{bmatrix} x(0) \\ x(0) \\ x(0) \\ x(0) \end{bmatrix} \\
&= \left[\begin{array}{cc} u(1) & u(2) \\ u(2) & u(1) \\ \hline u(1) & u(2) \\ u(2) & u(1) \end{array}\right] \begin{bmatrix} c_5^2 \\ c_5^4 \end{bmatrix} - j \left[\begin{array}{cc} v(1) & v(2) \\ v(2) & v(1) \\ \hline -v(1) & -v(2) \\ -v(2) & -v(1) \end{array}\right] \begin{bmatrix} s_5^2 \\ s_5^4 \end{bmatrix} \\
&\quad + \begin{bmatrix} x(0) \\ x(0) \\ x(0) \\ x(0) \end{bmatrix}
\end{aligned} \tag{8a}$$

$$\begin{aligned}
\boldsymbol{Y} &= \boldsymbol{u} \bullet \boldsymbol{c} - j\boldsymbol{v} \bullet \boldsymbol{s} + \boldsymbol{x}(\boldsymbol{0}) \\
&= (R(\boldsymbol{u}) \bullet \boldsymbol{c} + I(\boldsymbol{v}) \bullet \boldsymbol{s} + R(\boldsymbol{x}(\boldsymbol{0}))) \\
&\quad + j(I(\boldsymbol{u}) \bullet \boldsymbol{c} - R(\boldsymbol{v}) \bullet \boldsymbol{s} + I(\boldsymbol{x}(\boldsymbol{0})))
\end{aligned} \tag{8b}$$

where $u(i) = x(i) + x(N-i)$, $v(i) = x(i) - x(N-i)$, $R(\cdot)$is the real part, and $I(\cdot)$ is the imaginary part. Notice that the even part matrix $\boldsymbol{u}$ has the same upper and lower halves, and the odd part matrix $\boldsymbol{v}$ has the same absolute value of upper and lower halves, just with different signs. So, constant multiplications [e.g., $c_5^2(x(1)+x(4))$] appear in both $Y(i)$ and $Y(N-i)$, $i = 1, \cdots, (N-1)/2$. So only the unique constant multiplications in $\{Y(i),\ i = 1, \cdots, (N-1)/2\}$ need to be calculated. The output $\{Y(N-i),\ i = 1, \cdots, (N-1)/2\}$ can be obtained simultaneously by negating the intermediate results of $v \bullet s$ of $\{Y(i),\ i = 1, \cdots, (N-1)/2\}$. Thus, compared with (6c),

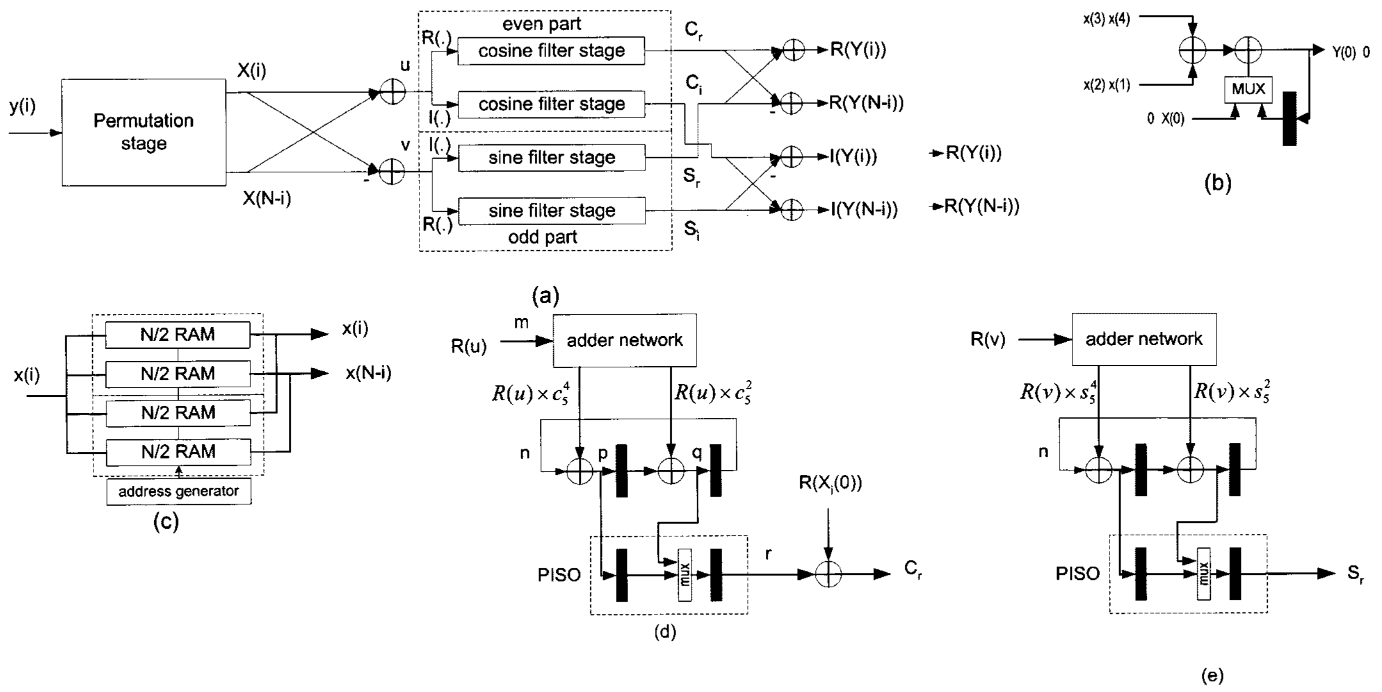


Fig. 3. (a) Proposed architecture design for the five-point DFT. (b) Separated calculation of $Yi(0)$. (c) Permutation stage. (d) Cosine filter stage. (e) Sine filter stage.

(8b) calculates two output values simultaneously but with similar hardware costs as (6c). We obtain two times the throughput at little overhead.

## C. Formulation for Nonprime-Length DFT

For nonprime-length DFT, e.g., power-of-two, we can use chirp-$z$ transform [15] to formulate $N$-point DFT into cyclic convolution form and then explore its symmetry properties. Chirp-$z$ transform can formulate any-length DFT into cyclic convolution but requires two extra multiplications, one for preprocessing, and one for postprocessing. However, for $N$ small, the extra multiplications can be reduced to hardwired multiplications. While $N$ is large, the area portion of the two extra multiplications will even be quite small compared to other computing blocks.

## IV. Architecture Design

Fig. 3(a) shows the proposed architecture for $N$-point DFT. This architecture implements the equation

$$\begin{aligned}\boldsymbol{Y} = {} & (R(\boldsymbol{u}) \bullet \boldsymbol{c} + I(\boldsymbol{v}) \bullet \boldsymbol{s} + R(\boldsymbol{x}(\boldsymbol{0}))) \\ & + j(I(\boldsymbol{u}) \bullet \boldsymbol{c} - R(\boldsymbol{v}) \bullet \boldsymbol{s} + I(\boldsymbol{x}(\boldsymbol{0}))). \quad (9)\end{aligned}$$

In the figure, $C_r = R(\boldsymbol{u}) \bullet \boldsymbol{c} + R(\boldsymbol{x}(\boldsymbol{0}))$, $C_i = I(\boldsymbol{u}) \bullet \boldsymbol{c} + I(\boldsymbol{x}(\boldsymbol{0}))$, $S_r = I(\boldsymbol{v}) \bullet \boldsymbol{s}$, and $S_i = R(\boldsymbol{v}) \bullet \boldsymbol{s}$. For illustration purposes, we take the five-point DFT as an example. The input data are first reordered to form cyclic convolution input. So the overall architecture is a transposed direct form for computing output $Y_i(1)-Y_i(4)$ for $i$th DFT computation. The output $Y_i(0)$ is generated by a separate accumulator shown in Fig. 3(b) since it is not in the cyclic formulation. In Fig. 3(b), at the first cycle we select input $X_i(0)$ and add it with $X_i(1)$ and $X_i(4)$. Then its result is accumulated with the sum of $X_i(2)$ and $X_i(3)$ to obtain $Y_i(0)$. The permutation stage shown in Fig. 3(c) is simply to reorder the input into sequence order $\{(X_i(1)X_i(4)), (X_i(2)X_i(3))\}$. This function can be easily constructed by four banks of RAM and address generators. The first two banks store the data used in the current block of computation, and the second two banks store the incoming data for the next block of computation. By enabling different banks for consecutive DFT computations, we can obtain the desired input without multiplexers.

After the permutation stage, the two input $X(i)$ and $X(N-i)$ are first combined together to generate $u$ and $v$, and then feed into the filter stages that generate the different constant multiplications. Fig. 3(d) and (e) shows the cosine filter stage that performs the multiplications with the cosine coefficients ($R(\boldsymbol{u}) \bullet \boldsymbol{c}$ and $I(\boldsymbol{u}) \bullet \boldsymbol{c}$), and the sine filter stage that performs the multiplications with the sine coefficients ($R(v) \bullet s$ and $I(v) \bullet s$), respectively. The addition of $X_i(0)$ only appears in the cosine stage. To save more hardware cost, the real and imaginary part of $u$ can be interleaved to enter one cosine filter stage, since the two cosine filter stages in the even part are the same. Similar method can be applied to the sine filter stage, too. The adder network in the filter stage can be derived as introduced in the previous section. The output of filter stage, $C_r$, $C_i$, $S_r$, and $S_i$ are added together as in (8a) to form the final output of the DFT.

Fig. 4 shows the function of the cosine filter stage in Fig. 3(d) by using the data flow diagram. The data flow of the sine filter stage is also similar, so we will not show it. In the first $(N-1)/2-1$ cycles of $i$th transform computation, the input $u_i$ is multiplied with constant cosine coefficients and accumulated with the results of previous input. Then the results are propagated through the delay chains. The first $(N-1)/2-1$ output of $q$ for each transform are routed back to the first tap to accumulate succeeding tap output. At the $(N-1)/2$th cycle of

| Time | m | n | p | q | r |
|---|---|---|---|---|---|
| 1 | $u_1(1)$ | 0 | $u_1(1)\times c_5^4$ | $u_1(1)\times c_5^2$ | 0 |
| 2 | $u_1(2)$ | $u_1(1)\times c_5^2$ | $u_1(1)\times c_5^2+u_1(2)\times c_5^4$ | $u_1(1)\times c_5^4+u_1(2)\times c_5^2$ | 0 |
| 3 | $u_2(1)$ | 0 | $u_2(1)\times c_5^4$ | $u_2(1)\times c_5^2$ | $u_1(1)\times c_5^4+u_1(2)\times c_5^2$ |
| 4 | $u_2(2)$ | $u_2(1)\times c_5^2$ | $u_2(1)\times c_5^2+u_2(2)\times c_5^4$ | $u_2(1)\times c_5^4+u_2(2)\times c_5^2$ | $u_1(1)\times c_5^2+u_1(2)\times c_5^4$ |
| 5 | $u_3(1)$ | 0 | $u_3(1)\times c_5^4$ | $u_3(1)\times c_5^2$ | $u_2(1)\times c_5^4+u_2(2)\times c_5^2$ |
| 6 | $u_3(2)$ | $u_3(1)\times c_5^2$ | $u_3(1)\times c_5^2+u_3(2)\times c_5^4$ | $u_3(1)\times c_5^4+u_3(2)\times c_5^2$ | $u_2(1)\times c_5^2+u_2(2)\times c_5^4$ |

Fig. 4. Data flow diagram in the Fig. 3(d), where $m$, $n$, $p$, $q$, and $r$ are the observing signals and the region.

TABLE I
HARDWARE COST COMPARISONS OF VARIOUS DESIGNS FOR COMPLEX INPUT

| | Adders | MUXs | Registers | RAM | DFFs in permutation | ROM | adder-network (adder numbers) |
|---|---|---|---|---|---|---|---|
| Proposed | 2N+11 | N+1 | 4N-2 | 4N-4 | 0 | 0 | (N-1)*2 for 8-b input<br>(N-1)*4 for 16-b input |
| Guo[8] | 6N-2 | 2N+10 | 10N-10 | 4N-4 | 4N-4 | $(N-1)*2^{L/2+2}$ | 0 |

Note: The above cost comparisons are all in terms of words. For comparison purpose, the cost of the adder network is estimated using the number of adders based on the results of [11].

each block of transform, since a new input sample of the next block of transform will come in, the intermediate results are parallel saved into parallel-in serial-output (PISO), as depicted in Fig. 3(d) and (e). Then the data in PISO are serially shifted out and accumulated with $X_i(0)$. Thus, we can get the DFT output by combining the output of filter stages. As shown in Fig. 4, there is no extra time between the adjacent data bundle for consecutive calculations. The hardware overhead paid for this is one extra addition, and registers and multiplexers in PISO that are linearly increased with the DFT length. Though DFT is being formulated as a cyclic convolution form, the block input, instead of the continuous input of DFT, results in a nonoverlapped computation and efficiency loss. To prevent such loss, the PISO is added to the filter stages for overlapped computation of adjacent blocks. Extension of the proposed design to DHT is straightforward, since both DFT and DHT share the similar kernels. By adding the real and imaginary DFT outputs of Fig. 3(a), we can obtain the DHT output. Similar designs can also be applied to other prime length DFT, DCT and DST by using different network designs.

## V. PERFORMANCE ANALYSIS AND COMPARISONS

By carefully examining the architecture shown in Fig. 3, we can summarize their hardware cost in Table I for various length $N$ (excluding the address generator). For real input, the hardware cost is half of that in Table I. For comparison, we also summarize the hardware cost of the memory-based design [8] in the Table I, since memory-based design also uses cyclic convolution approach and is similar to our design. As shown in the table, the proposed design has lower hardware cost. The hardware savings are due to the combination of cyclic formulation, subexpression sharing and symmetry exploration. Also, the I/O channels are limited to the extreme ends of the architectures as that in [8].

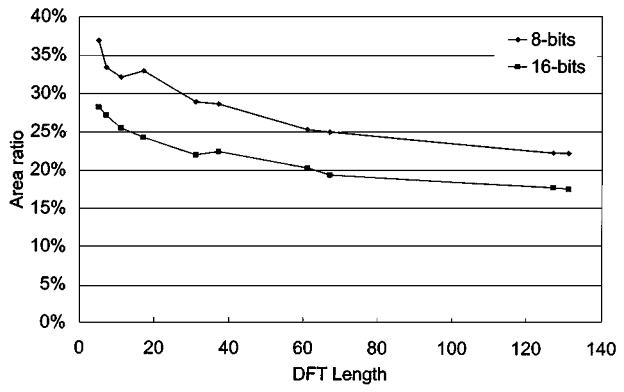


Fig. 5. Gate-area ratio of proposed design versus the memory-based design [8] for different word length and transform length. The word length of coefficient and input is the same for each curve.

The number of cycles to complete one DFT is $(N-1)/2$, half of that in [8]. The cycle time is limited by the $T_{\text{adder-network}} + T_{\text{addition}} + T_{\text{DFF-setup}} + T_{\text{DFF-hold}}$. $T_{\text{adder-network}}$ depends on the subexpression sharing results of coefficients. Usually, no more than $\log_2(W_c)$ stages of $T_{\text{addition}}$ are required, where $W_c$ is the wordlength of the coefficients. This cycle time is smaller than that in [8]. Besides, the throughput of the proposed design is two times that in [8], since two output values are calculated at a time.

To illustrate the advantages of the proposed design, we weight these hardware items with their equivalent gate counts [17]. Fig. 5 shows the gate-area ratio plot of the proposed design versus the memory-based design [8]. In the figure, the adder numbers used in the adder network are derived for each transform length and word length. Those numbers are very close to the estimated value in Table I. For gate-count estimation of ROM and RAM, we use one gate count area =345.6 $\mu$m$^2$. The costs of adder, multiplexer (MUX), and $D$ flip-flop (DFF) are

TABLE II
COMPARISONS OF SEVERAL DFT DESIGNS. ALL ARE FOR REAL NUMBER COMPUTATIONS

| Designs | multiplier | adder | registers | Throughput |
|---|---|---|---|---|
| Fang[4] | 2N | 4N | 8N | 1/N |
| Chan[5] | 4N | 4N | 6N | 1/N |
| Murthy[6] | 4N | 4N | 6N | 1/N |
| Chan[7] | 4N | 4N | 8N | 1/N |
| Proposed | 0 | 6N+7 | 4N-2 | 2/N |

seven gates, three gates, three gates, and eight gates, respectively. As shown in the figure, the hardware saving of the proposed design gradually increases as the DFT length increases. For example, in the case of a 16-bit coefficient for $N$ =61, the proposed design can save up to 80% of gate area. If we compare the two designs at the same throughput, the proposed design can save up to 90% of gate area due to two times the throughput. The gate count of the proposed design for $N = 61$, and 16-bit coefficient case is estimated to be 99 760, including the RAM cost. The interconnection area does not have strong impact in modern IC process with five or more interconnection layers. The speed of the above case is estimated to be 5.12 ns, which is smaller than the designs in [8] (8.5 ns), and a 16 $\times$ 16 multiplier (21.31 ns), in other multiplier-additions based designs.

Other $N$-point DFT systolic array architectures often require N or more processors. Each processor requires one complex multiplication–addition (CMAC), which turns out to be $4 * N * (W_x + 1)$ additions ($W_x$ is the word length of the input). These CMAC-based designs have higher hardware cost than our proposed design due to variable complex CMAC instead of the adder-only design in our approach. The complex multiplications result in very high hardware cost and long cycle time. The design in [5] requires $N$ CMAC processors and each processor requires one external coefficient input that results in high I/O bandwidth. Designs in [6]–[7] keep I/O channels in the terminals but also require $N$ CMAC processors. The design in [4] explores the symmetry of DFT but still requires $N/2$ complex multipliers and $N$ complex adders, which is still higher than the proposed design. The drawback of the above designs is that they propagate the constant DFT coefficients. Table II lists a comparison of these designs. In this table, we include the adder network cost (16-bit case) in the adder cost for the proposed design. We only list the adder and register cost since they occupy most of the area. From the table, we can easily find that CMAC-based designs require larger area cost than ours. The design in [17] uses cyclic convolution and bit-level implementation, but no sharing is exploited in their designs. So, it also requires more area cost.

To summarize, the proposed architecture has several distinctive features. First, due to cyclic formulation and filter-based design, the I/O channels of the architecture are in the extreme ends of the architecture that results low I/O bandwidth. Second, since word-level sharing exploits the symmetries of DFT coefficients, two times the throughput can be obtained with little overhead. Third, all the constant multiplications are efficiently realized by only shift operations and additions, which are minimized by the subexpression sharing technique. Thus, we can save up to 80% of area cost and attain low hardware costs. Our method results in faster speed and better power efficiency due to a saving of redundant computations and sharing common computation.

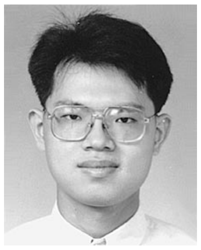
## VI. CONCLUSION

In this paper, we propose a hardware-efficient architecture for the DFT designs. The proposed architecture combines the advantages of cyclic convolution, word-level sharing, and bit-level subexpression sharing. The transform equation is reformulated with cyclic convolution, such that subexpression sharing can be efficiently applied to save hardware costs. Symmetry properties of DFT coefficients are explored to enhance the throughput by a factor of two. Comparing to the previous memory-based design [8], the presented approach can save up to 80% of gate count with two times the throughput for length $N$ =61. This design method can be readily applied to other prime-length DFT, DCT, and DHT designs. A restriction of the proposed DFT design is that it only deals with prime-length DFTs. This restriction should not put a severe limitation on the DFT because we can append zeros to a nonprime-length input sequence to attain a prime length. This operation affects the energy of the output but has no effect on its shape. Another approach for nonprime-length DFT is to use the cyclic formulation by the Chirp-$z$ transform method [15]. It can be used together with the proposed approach for an efficient implementation. However, the Chirp-$z$ transform method will introduce additional pre-and post-multiplications. Other nonprime length DCTs, such as 1-D eight-point DCTs, can also use this approach in implementation by reformulating it into cyclic convolution [18].

## REFERENCES

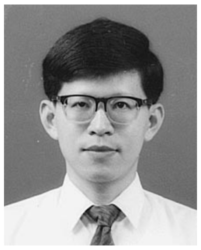
[1] H. T. Kung, “Why systolic architectures?,” *Comput. Mag.*, vol. 15, pp. 37–45, Jan. 1982.

[2] J. A. Beraldin, T. Aboulnasr, and W. Steenart, “Efficient one-dimensional systolic array realization of discrete Fourier transform,” *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 36, pp. 1665–1667, Oct. 1988.
[3] ——, “Efficient one-dimensional systolic array realization of the discrete Fourier transform,” *IEEE Trans. Circuits Syst.*, vol. 36, Jan. 1989.
[4] W. H. Fang and M. L. Wu, “An efficient unified systolic architecture for the computation of discrete trigonometric transforms,” *Proc. ISCAS*, vol. 3, pp. 2092–2095, 1997.
[5] L. W. Chan and M. Y. Chen, “A new systolic array for discrete Fourier transform,” *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 36, pp. 1665–1666, Oct. 1988.
[6] N. R. Murthy and M. N. S. Swamy, “On the real-time computation of DFT and DCT through systolic architectures,” *IEEE Trans. Signal Processing*, vol. 42, pp. 988–991, Apr. 1994.
[7] E. Chan and S. Panchanathan, “A VLSI architecture for DFT,” *Proc. 36th Midwest Symp. Circuits and Systems*, vol. 1, pp. 292–295, 1993.
[8] J. I. Guo, C.-M. Liu, and C.-W. Jen, “The efficient memory-based VLSI array designs for DFT and DCT,” *IEEE Trans. Circuits Syst. II.*, vol. 39, pp. 723–733, Oct. 1992.
[9] T. S. Chang and C.-W. Jen, “Hardware efficient transform designs with cyclic formulation and subexpression sharing,” *Proc. ISCAS*, vol. 2, pp. 398–401, 1998.
[10] D. R. Bull and D. H. Horrocks, “Primitive operator digital filters,” *IEE Proceedings-G*, vol. 138, pp. 401–411, June 1991.
[11] A. G. Dempster and M. D. Macleod, “Use of minimum-adder multiplier blocks in FIR digital filters,” *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 569–577, Sept. 1995.
[12] M. Potkonjak *et al.*, “Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination,” *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 151–165, Feb. 1996.
[13] R. I. Hartley, “Subexpression sharing in filters using canonic signed digit multipliers,” *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 677–688, Feb. 1996.
[14] C. M. Radar, “Discrete Fourier transforms when the number of data samples is prime,” *Proc. IEEE*, vol. 56, pp. 1107–1108, 1968.
[15] C. S. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms*. New York: Wiley, 1985.
[16] Compass, *PASSPORT library, 0.6 micron 5-volt high performance standard cell library*, 1996.
[17] S. Gudvangen and A. G. J. Holt, “Computation of prime factor DFT and DHT/DCCT algorithms using cyclic and skew-cyclic bit-serial semisystolic IC convolvers,” *Proc. Inst. Elect. Eng.—G*, vol. 137, no. 5, pp. 373–389, Oct. 1990.
[18] Y.-H. Chan and W.-C. Siu, “A cyclic correlated structure for the realization of the discrete cosine transform,” *IEEE Trans. Circuits Syst. II*, vol. 39, pp. 109–113, Feb. 1992.



**Tian-Sheuan Chang** (S’93–M’00) received the B.S., M.S., and Ph.D. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 1993, 1995, and 1999, Hsinchu, Taiwan.

He is currently a Principle Engineer at Global Unichip Co., Hsinchu, Taiwan. His research interests includes VLSI design, digital signal processing, and computer architecture.



**Jiun-In Guo** was born in Kaohsiung, Taiwan, R.O.C., in 1966. He received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1989 and 1993, respectively.

He is currently an Associate Professor of the Department of Electronics Engineering, National Lien-Ho Institute of Technology, Miaoli, Taiwan. He was the Director of the Department of Electronics Engineering, National Lien-Ho Institute of Technology, from 1996 to 1999. His research interests include image, speech, and digital signal processing, VLSI architecture design, and VLSI implementation.

**Chein-Wei Jen** (S’78–M’87) received the B.S. degree from National Chiao Tung University, Hsinchu, Taiwan, in 1970, the M.S. degree from Stanford University, Stanford, CA, in 1977, and the Ph.D. degree from National Chiao Tung University in 1983.

He is currently with the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, as a Professor. During 1985–1986, he was with the University of Southern California at Los Angeles as a Visiting Researcher. His current research interests include VLSI design, digital signal processing, processor architecture, and design automation. He has held four patents and published over 30 journal papers and 70 conference papers in these areas.

Dr. Jen was a Program Committee member of ICCD’94 and ICCE’95–97. He received the 1990 Best Paper Award from the Engineer Society, and the 1994 and 1995 Best Paper Awards of the HD-Media Conference. He is a member of Phi Tau Phi.