CONFLICT-FREE PARALLEL MEMORY ACCESS SCHEME FOR FFT PROCESSORS

Jarmo H. Takala, Tuomas S. Järvinen, and Harri T. Sorokin

Tampere University of Technology, P.O.B. 553, FIN-33101 Tampere, Finland

ABSTRACT

In this paper, a parallel access scheme for constant geometry FFT algorithms is proposed, which allows conflict-free access of operands distributed over parallel memory modules. The scheme is a linear transformation and the address generation is performed with the aid of bit-wise XOR operations. Different FFT lengths can be supported with the aid of a simple address rotation unit. The scheme is general supporting several radices in FFT computations and different numbers of parallel memory modules. The scheme allows parallel butterfly computations independent of the FFT length.

1. INTRODUCTION

Due to the symmetric structure of Cooley-Tukey fast Fourier transform (FFT) algorithm, it lends itself to VLSI implementations and several realizations with varying level of parallelism have been proposed over the years. In parallel implementations, memory bandwidth may limit the performance of the system; in radix-S FFT algorithm, the butterfly operation consumes S operands and produces S results, thus S memory accesses should be performed in parallel. Often the memory bandwidth is increased by partitioning the memory into S independent memory modules, which can be accessed simultaneously. Such a memory architecture is referred to as an interleaved memory system and it is illustrated in Fig. 1. If the FFT architecture contains d radix-S butterfly units, dS parallel memory modules are needed.

Often the operands to be accessed simultaneously lie in the same memory module thus the parallel access cannot be performed. In FFT algorithms, such conflicts are due to the data reordering between the computational columns as seen in Fig. 2 where signal flow graphs of 16-point constant geometry algorithms for radix-2 and radix-4 FFT are shown. Therefore, the principal problem in interleaved memory systems is to find a method to distribute data over the memory modules in such a way that the conflicts are avoided.

The conflict-free access can arranged with the aid of double-buffering but this is extremely expensive when long FFTs are to be computed. In order to minimize the memory consumption, the results of computation should be written into the same memory locations where the operands were read. In [1], it was found that in radix-2 FFT the operands can be distributed over two memory modules based on the parity of the operand index. This observation was exploited in [2] where an address generator for 2-memory radix-2 FFT systems were proposed. In [3], the address generation was simplified at the expense of additional registers in the interconnection network. The registers are used to delay the conflicting write accesses. In [4], the conflicts were avoided by allocating an additional memory module for delaying certain write accesses. The previous solutions covered only radix-2 FFTs and assume that a single radix-2 butterfly is computed at a time.

A general solution for radix-S FFTs was given in [5] but again it was assumed that the computations are performed a single butterfly at a time. Memory addressing in FFT systems containing parallel butterfly computations is considered in [6] where the access conflicts are avoided by reordering the operands in an interconnection network. The reordering is performed in time and space, which requires additional operand registers.

The memory distribution in FFT computation has also been considered in supercomputing area. In [7], an access scheme based on linear transformation is proposed but the described address generation is complex requiring binary matrix multiplication. A far more simple scheme was proposed in [8]. However, access conflicts cannot be completely avoided in these schemes.

In this paper, a parallel access scheme is proposed, which allows conflict-free access of operands distributed over the parallel memories. The scheme supports several radices in FFT computations and it allows parallel butterfly computations independent of the FFT length.

2. PRELIMINARIES

FFT algorithms can be scheduled into a form where the interconnections between the processing columns of the signal flow graph are stride permutations. In general, interconnections in radix-2^s FFT algorithm are related to stride-by-2^s permutation. Such permutations can be described with the aid of matrix transpose; stride-by-S permutation of an N-element vector can be performed by dividing the vector into S-element sub vectors, organizing them into $S \times (N/S)$ matrix form, transposing the obtained matrix, and rearranging the result back to the vector representation [9]. Another interpretation is to use indexing functions as used in the following formal definition.

Definition 1 (Stride Permutation) *Let us assume a vector* $X = (x_0, x_1, \ldots, x_{N-1})$. *Stride-by-*$S$ *permutation reorders* X *as* $Y = (x_{f_{N,S}(0)}, x_{f_{N,S}(1)}, \ldots, x_{f_{N,S}(N-1)})^T$ *where the index function* $f_{N,S}(i)$ *is given as*

$$f_{N,S}(i) = (iS \bmod N) + \lfloor iS/N \rfloor \mid N \operatorname{rem} S = 0,\ i = 0, 1, \ldots, N-1 \quad (1)$$

where mod *is the modulus operator,* $\lfloor \cdot \rfloor$ *is the floor function, and* rem *is remainder.*

The stride permutation of an array X can also be expressed in matrix form as $Y = P_{N,S}X$ where $P_{N,S}$ is stride-by-S permutation matrix of order N.

In this paper, we limit ourselves to practical cases where array lengths and strides are powers of two, $N = 2^n$, $S = 2^s$. A property of stride permutations in such cases is given in the following.

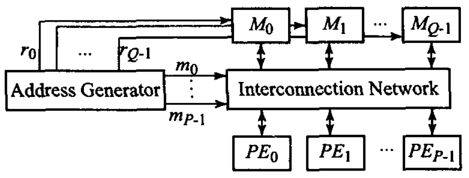


Figure 1: Interleaved memory system. PE$_i$: Processing element. M$_i$: Memory module. m_i: Module address. r_i: Row address.

Theorem 1 (Factorization of stride permutations) *Let $ab \le N$, then*

$$P_{N,ab} = P_{N,a}P_{N,b} = P_{N,b}P_{N,a} \qquad (2)$$

The proof for the previous theorem can be found, e.g., from [9].

Let us assume that an N-element array is to be distributed over Q independent memory modules, $N = 2^n, Q = 2^q$. In such a case, an access scheme performs two mappings; the index address of an element, $a = (a_{n-1}, a_{n-2}, \ldots, a_0)^T$, is mapped onto a module address, $m = (m_{q-1}, \ldots, m_0)^T$, and a row address, $r = (r_{n-q-1}, \ldots, r_0)^T$. It should be noted that in this representation the least significant bit of a is in the bottom of the vector. If the access scheme is a linear transformation, the address arithmetic is based on modulo-2 arithmetic, which implies that the arithmetic is realized with bit-wise XOR operations. The address mappings in the linear transformation can be expressed with binary transformation matrices as

$$r = Ka\,; \quad m = Ta \qquad (3)$$

The matrices K and T are the row and module transformation matrix, respectively. Often K is defined as

$$K = \left(I_{n-q} 0_{(n-q),q}\right) \qquad (4)$$

where I_k denotes the identity matrix of order k and $0_{i,k}$ denotes an $i \times k$ matrix of zeros. Then the row address is obtained simply by extracting the $(n-q)$ most significant bits of the address a.

3. CONFLICT-FREE ACCESS SCHEME FOR STRIDE PERMUTATION

Let us first investigate the read and write accesses in radix-2 FFT illustrated in Fig. 2(a). In order to minimize memory consumption, the results should be stored into the same memory locations where the operands were obtained. In the first iteration, operands are read in linear order, i.e., according to $P_{16,1}$. The results need also to be written in linear order, although they should be permuted according to stride-by-2, $P_{16,2}$. This implies that, in the second iteration, the operands should be read and written according to $P_{16,2}$. In the third iteration, the accesses are according to $P_{16,4}$, which is due to Theorem 1. Eventually we find that $\log_2 N$ different strides are needed, i.e., all the strides of power-of-two from 1 to $N/2$.

In [10], it was shown that an access scheme supporting several strides cannot be designed for matched memory systems, if the access should be conflict-free regardless of the array length and initial address. A matched memory system refers to an organization where the number of memory modules is the same as the number of operands accessed in a cycle by the processing elements. We may, however, relax the requirements. We make the following assumptions: a) the array length is constant and power-of-two, $N = 2^n$, b) the

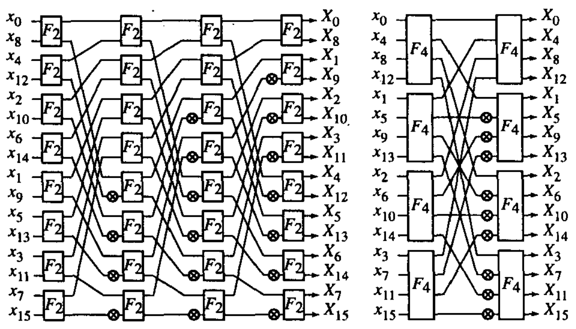


Figure 2: Signal flow graphs of FFT algorithms: (a) radix-2 and (b) radix-4 algorithm. F_k: k-point FFT.

array is stored in 2^n-word boundaries, c) the number of memory modules is a power-of-two, $Q = 2^q$, and d) the strides in stride permutation access are powers-of-two, $S = 2^s$. Assumption a) implies that constraints on the initial address need to be set resulting in the assumption b). Such an constraint has already been used in several commercial DSP processors for performing circular addressing. Assumption c) is actually practical in digital systems. Assumption d) implies that the address mapping should produce a q-bit memory module address and a $(n-q)$-bit row address. All these assumptions can be considered practical.

3.1. Access Scheme

We propose an access scheme, which can be used under the previous assumptions. The row and module address mappings are defined as follows

$$\begin{aligned} r_i &= a_{i+q}, i = 0, 1, \ldots, n-q-1; \\ m_i &= \bigoplus_{k=0}^{l_{n,q}(i)} a_{(jq+i) \bmod n}, \; i = 0, 1, \ldots, q-1\,; \\ l_{n,q}(i) &= \lfloor (n + q - \gcd(q, n \bmod q) - i - 1)/q \rfloor \end{aligned} \qquad (5)$$

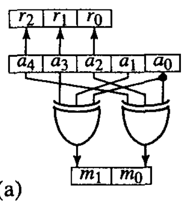
where $\oplus$ is bit-wise XOR operation and $\gcd(\cdot)$ is the greatest common denominator.

The module transformation is dependent on the array length N and the number of memory modules Q, thus we introduce a new notation for the module transformation matrix: $T_{N,Q}$. The module transformation matrix $T_{32,4}$ is the following

$$T_{32,4} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \qquad (6)$$

The contents of the memory modules stored according to $T_{32,4}$ is illustrated in Fig. 3(b) and the module address can be computed with the circuit shown in Fig. 3(a). All the possible parallel accesses in this case are listed in the following:

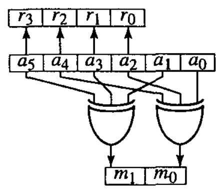
$P_{32,1}$	:	$([0,1,2,3],[4,5,6,7],\ldots,[28,29,30,31])$
$P_{32,2}$	:	$([0,2,4,6],[8,10,12,14],\ldots,[25,27,29,31])$
$P_{32,4}$	:	$([0,4,8,12],[16,20,24,28],\ldots,[19,23,27,31])$
$P_{32,8}$	:	$([0,8,16,24],[1,9,17,25],\ldots,[7,15,23,31])$
$P_{32,16}$	:	$([0,16,1,17],[2,18,3,19],\ldots,[14,30,15,31])$



(a)

m 0	1	2	3
0	3	2	1
7	4	5	6
10	9	8	11
13	14	15	12
19	16	17	18
20	23	22	21
25	26	27	24
30	29	28	31

(b)



(c)

m 0	1	2	3
0	1	2	3
5	4	7	6
10	11	8	9
15	14	13	12
17	16	19	18
20	21	22	23
27	26	25	24
30	31	28	29
34	35	32	33
39	38	37	36
40	41	42	43
45	44	47	46
51	50	49	48
54	55	52	53
57	56	59	58
60	61	62	63

(d)

Figure 3: Access scheme for 4-module system: (a) module address generation and (b) contents of memory modules for transform matrix in (6) and (c) module address generation and (d) contents of memory for matrix in (7).

It can be seen that all the possible stride permutation accesses are conflict-free. The transformation matrix $T_{64,4}$ would be

$$T_{64,4} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{7}$$

and the module address generation is shown in Fig. 3(c) and the storage is depicted in Fig. 3(d). Once again, all the possible power-of-two stride accesses are conflict-free.

3.2. Validation

The presented access scheme has been validated with the aid of computer simulations by generating several storage organizations and verifying that each access is conflict-free. For a given array length $N = 2^n$, the number of memory modules Q was varied to cover all the possible numbers of powers-of-two, i.e., $Q = 2^0, 2^1, \dots, 2^{n-1}$. For each parameter pair (N, Q), all the stride permutation accesses were performed with strides covering all the possible powers-of-two: $S = 2^0, 2^1, \dots, 2^{n-1}$ and each parallel access was verified to be conflict-free. The power-of-two array lengths were iterated from 2^1 to 2^{20}. The extensive simulation showed that the presented access scheme provides conflict-free parallel stride permutation access in practical cases, i.e., array lengths up to 2^{20}, for all the possible power-of-two strides on matched memory systems where the number of memory modules is a power-of-two.

$$T_{32,16} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{64,16} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{128,16} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$T_{256,16} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 4: Module transformation matrices in 16-memory module system.

3.3. Address Generation

Before going into implementations, we may investigate the structure of $T_{N,Q}$ when N is varied. In practical systems, the number of memory modules , Q, is constant; Q is only a design time parameter. As an example transformation matrices for 64-module systems are illustrated in Fig. 4 and few observations can be made from the structure of these matrices. The off-diagonal ones affect at most the $q - 1$ least significant bits of the address a as defined in (5). In addition, the structure of off-diagonals depends on the relation between n and q but in practical systems q is constant, thus the structure depends only on the array length. However, there are only q different structures; the off-diagonal structure has periodic behavior when the array length is increasing. In Fig. 4, one complete period is shown and $T_{512,16}$ would have the same off-diagonal structure as $T_{32,12}$. The structure of off-diagonals implies that several array lengths can be supported if a predetermined control word configures additional hardware to perform the functionality of the off-diagonals. Such a configuration is actually simple by noting that the form of off-diagonals in different array lengths indicates rotation of the least significant bits in a. The number of bits rotated is dependent on the relation between n and q.

According to the previous observations, the computation of the module address m can be interpreted as follows. First, the address a is divided into q-bit fields, F^i, starting from the least significant bit of a. If $e = n \bmod q > 0$, the e most significant bits of a exceeding the q-bit block border are extracted as a bit vector L. Next, a q-bit field X is formed by extracting the $(q - \gcd(q, e))$ least significant bits of the address a and placing zeros to the most significant bits. The bit vector X is rotated $g = (n - q \bmod q)$ bits to the left to obtain a bit vector O. Finally the module address m is obtained by performing bit-wise XOR operation between the vectors F_i, L, and O. A principal block diagram of the module address generation according to the previous interpretation is illustrated in Fig. 5(a). This block diagram contains a rotation unit shown in Fig. 5(b), which computes the vector O.

The main advantage of the presented scheme can be seen from the block diagram in Fig. 5. In the address generation, each individual XOR is performed on at most $\lfloor n/q \rfloor + 2$ bit lines while

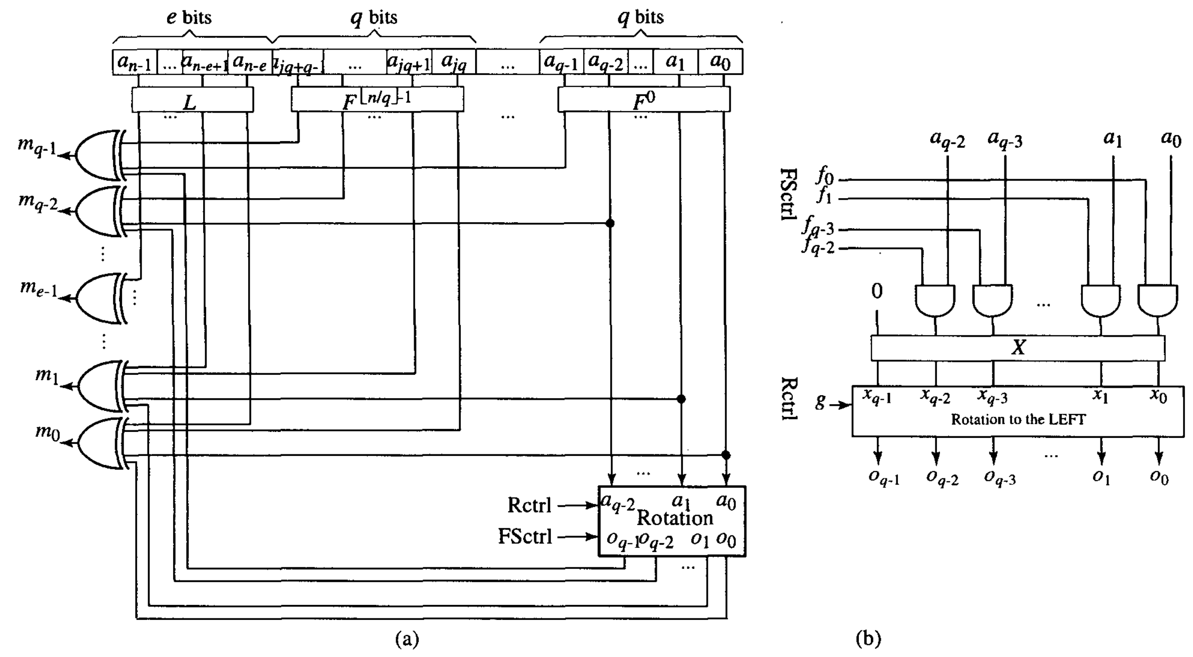


Figure 5: Principal block diagram of (a) module address generation and (b) rotation unit. Rctrl: Rotation control. FSctrl: Field selection control.

in other schemes, e.g., in [7], some XORs require all the n address bits, which complicates implementations when several array lengths need to be supported. The support for different array lengths requires only a single predetermined control word, which defines the bit selection and rotation. This control word needs to be modified only when the array length is changed.

4. CONCLUSIONS

In this paper, a conflict-free stride permutation access scheme for constant geometry FFT computations was presented. It was assumed that 2^n data elements are distributed over 2^q independent memory modules. The performed simulations showed that all the possible power-of-two stride permutation accesses are conflict-free. The module address generation is simple requiring only bit-wise XOR operations. It was shown that several array lengths can be supported by including a q-bit rotation into the module address generator. In this case, all the additional operations are performed on the $q-1$ least significant bits of the index address.

5. REFERENCES

[1] M. C. Pease, "Organization of large scale Fourier processors," *J. Assoc. Comput. Mach.*, vol. 16, no. 3, pp. 474–482, July 1969.

[2] D. Cohen, "Simplified control of FFT hardware," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 24, no. 6, pp. 255–579, Dec. 1976.

[3] Y. Ma, "An effective memory addressing scheme for FFT processors," *IEEE Trans. Signal Processing*, vol. 47, no. 3, pp. 907–911, Mar. 1999.

[4] C.-H. Chang, C.-L. Wang, and Y.-T. Chang, "A novel memory-based FFT processor for DMT/OFDM applications," in *Proc. IEEE ISCAS*, Orlando, FL, U.S.A., May 30 –June 2 1999, vol. 4, pp. 1921–1924.

[5] L. G. Johnson, "Conflict free memory addressing for dedicated FFT hardware," vol. 39, no. 5, pp. 312–316, May 1992.

[6] J. A. Hidalgo, J. Lopéz, F. Argüello, and E. L. Zapata, "Area-efficient architecture for fast Fourier transform," *IEEE Trans. Circuits Syst. II*, vol. 46, no. 2, pp. 187–193, Feb. 1999.

[7] A. Norton and E. Melton, "A class of boolean linear transformations for conflict-free power-of-two stride access," in *Proc. Int. Conf. Parallel Processing*, St. Charles, IL, U.S.A., Aug. 17–21 1987, pp. 247–254.

[8] D. T. Harper III, "Block, multistride vector, and FFT accesses in parallel memory systems," *IEEE Trans. Parallel and Distrib. Syst.*, vol. 2, no. 1, pp. 43–51, Jan. 1991.

[9] J. Granata, M. Conner, and R. Tolimieri, "Recursive fast algorithms and the role of the tensor product," *IEEE Trans. Signal Processing*, vol. 40, no. 12, pp. 2921–2930, Dec. 1992.

[10] D. T. Harper III, "Increased memory performance during vector accesses through the use of linear address transformations," *IEEE Trans. Comput.*, vol. 41, no. 2, pp. 227–230, Feb. 1992.