The Radix-*r* One Stage FFT Kernel Computation

Marwan A. Jaber and Daniel Massicotte

Université du Québec à Trois-Rivières, Electrical and Computer Engineering Department
Laboratory of Signal and System Integrations, www.uqtr.ca/lssi
C.P. 500, Trois-Rivières, Québec, Canada, G9A 5H7
{marwan.jaber, daniel.massicotte}@uqtr.ca

Abstract– The FFT process is an operation that could be performed through different stages. In each stage, the butterfly operation is computed in which the accessed data is multiplied by certain W^{α}, added or subtracted and finally it is stored or held for further processing. This process is repeated to each stage until the final stage where the processed data is driven to the output. In this paper, an appropriate indexing or mapping schemes between the input data and the coefficient multipliers throughout the different stages are yield to a computation single stage by collapsing all stages into a computation single stage. The result is a reduction of communication load and arithmetic operations.

Index Terms– Discrete Fourier transforms, Frequency domain analysis, Parallel processing

1. Introduction

The Discrete Fourier Transform (DFT) is a fundamental digital signal-processing algorithm used in many applications, including frequency analysis and frequency domain processing, such as speech compression, meanwhile the frequency domain processing allows for the efficient computation of the convolution integral (for linear filtering) and of the correlation integral (for correlation analysis), and in wireless communication system based on Orthogonal frequency division multiplex (OFDM), which the FFT is an operator key [4].

The definition of DFT is shown in equation (1), $x[n]$ is the input sequence, $X[k]$ is the output sequence, N is the transform length and w_N is the N^{th} root of unity ($w_N = e^{-j2\pi/N}$). Both $x[n]$ and $X[k]$ are complex valued sequences.

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k \in [0, N-1] \tag{1}$$

DFT is the decomposition of a sampled signal in terms of sinusoidal (complex exponential) components, and because of its computational requirements, the DFT algorithm, which requires N^2 complex value multiplication plus a smaller number of operations to complete a complex value addition or subtraction, usually is not used for real time signal processing. Several efficient methods have been developed to compute the DFT, "Cooley and Tukey presented their approach showing a number of multiplications required to compute the DFT of a sequence may be considerably reduced to $N\log_2 N$ by using one of the fast Fourier transform (FFT) algorithms [1]". One of the bottlenecks in most applications, where high performance is required, is the FFT/IFFT processor.

In this paper, the structure of the one stage algorithm for the dedicated FFT will be elaborated. The main objective of this proposal is reduction in communication load, reduction in computation and particularly reduction in the number of multiplications. The advantage of appropriately breaking the DFT in terms of its partial DFTs is that the number of multiplications and the number of stages may be controlled. The number of stages often corresponds to the amount of global communication and/or memory accesses in implementation, and thus, reduction in the number of stages is beneficial. Minimizing the computational complexity may be done at the algorithmic level of the design process, where the minimization of operations depends on the number representation in the implementation. Minimizing the communication load is achieved on the architecture level, where issues like possibility to power down Despite of the Cooley-Tukey's clear definition stating that the DFT is a combination of its partial DFTs, researchers used to express the DFT in terms of its partial DFTs as:

$$X[k] = \sum_{n=0}^{(N/r)-1} x[rn] w^{rnk} + \cdots + \sum_{n=0}^{(N/r)-1} x[rn+(r-1)] w^{(rn+(r-1))k} \tag{2}$$

As a result the mathematical representation of the DFT into its partial DFTs is not well defined yet. The problem resides in finding the mathematical model of the combination phase, in which the concept of butterfly computation should be well structured in order to obtain the right mathematical model.

The paper is organized as follows; in Section 2 a butterfly operation is defined. Section 3 is devoted to describe in details the proposed FFT method and the modified radix-*r* FFT. The implementation aspects are given to Section 4, while Section 5 draws conclusions.

2. The Butterfly Processing Element

The basic operation of a radix-*r* BPE is the so-called butterfly in which *r* inputs are combined to give the *r* outputs via the operation [2]:

$$\mathbf{X} = \mathbf{B}_r \mathbf{x} \tag{3}$$

where $\mathbf{x} = [x[0], x[1], \ldots, x[r-1]]^T$ is the input BPE vector and $\mathbf{X} = [X[0], X[1], \ldots, X[r-1]]^T$ is the output BPE vector. $\mathbf{B}_r$ is the butterfly matrix, dim($\boldsymbol{B}_r$)=$r\times r$, which can be expressed as

$$\mathbf{B}_r = \mathbf{W}_N^r \mathbf{T}_r \tag{4}$$

for the decimation in frequency (DIF) process, and

$$\mathbf{B}_r = \mathbf{T}_r \mathbf{W}_N^r \tag{5}$$

for the decimation in time (DIT) process. In both cases, DIT and DIF, the twiddle factor matrix, $\mathbf{W}_N$, is a diagonal matrix defined by $\mathbf{W}_N = \operatorname{diag}\left(1, W_N^p, W_N^{2p}, \ldots, W_N^{(r-1)p}\right)$ with $p = 0, 1, \ldots, \log_r N - 1$ and $\mathbf{T}_r$ is the *adder-tree* in the butterfly structure

$$\mathbf{T}_r = \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^{N/r} & W_N^{2N/r} & \cdots & W_N^{(r-1)N/r} \\ W_N^0 & W_N^{2N/r} & W_N^{4N/r} & \cdots & W_N^{2(r-1)N/r} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{(r-1)N/r} & W_N^{2(r-1)N/r} & \cdots & W_N^{(r-1)^2 N/r} \end{bmatrix} \tag{6}$$

where dim($\mathbf{T}_r$)=$r \times r$. We defined $\left[\mathbf{T}_r\right]_{l,m}$ the element of the lth line and mth column of the matrix $\mathbf{T}_r$.

We rewrite (6) as

$$\left[\mathbf{T}_r\right]_{l,m} = W_N^{[\![(lmN/r)]\!]_N}, \qquad (7)$$

with l=0,1,…,r-1, m=0,1,…,r-1 and $[\![\mathrm{x}]\!]_N$ represents the operation x modulo N.

If we pay attention to the elements of the adder-tree matrix $\mathbf{T}_r$ and to the elements of the twiddle matrix $\mathbf{W}_N$, we could notice that both of them contain twiddle factors. So, by controlling the variation of the twiddle factor during the calculation of a complete FFT, results in incorporating the twiddle factors and the adder matrix in a single-stage of calculation.

According to equation (3), $\mathbf{B}_r$ is the product of the twiddle factor matrix $\mathbf{W}_N$ and the adder-tree matrix $\mathbf{T}_r$.

By defining $\mathbf{W}_{N\,(u,v,s)}$ the set of the twiddle factor matrix $\mathbf{W}_N^r$ as:

$$\mathbf{W}_{N\,(u,v,s)} = \mathrm{diag}\left(W_{N\,(0,v,s)}, W_{N\,(1,v,s)}, \cdots, W_{N\,(r\text{-}1,v,s)}\right), \qquad (8)$$

with the indices $u = 0,1,\ldots,r-1$, $v = 0,1,\ldots,V-1$, and $s = 0,1,\ldots,S-1$ where r is the radix-r, V is the number of words, $V = N/r$, and S is the number of stages (or iteration), $S = \log_r N$. From (8), we can define all matrix elements as (lth line, mth column)

$$\left[\mathbf{W}_N\right]_{l,m\,(u,v,s)} = \begin{cases} W_N^{[\![\lfloor v/u^s \rfloor l\, u^s]\!]_N} & \text{for } l = m \\ 0 & \text{elsewhere} \end{cases}, \qquad (9)$$

l=0,1,…,r-1, m=0,1,…,r-1 and $\lfloor \mathrm{x} \rfloor$ defined the integer part operator of x. Therefore, the modified radix-r butterfly computation $\mathbf{B}_r$ will be expressed:

$$\mathbf{B}_r = \mathbf{W}_{N\,(u,v,s)} \mathbf{T}_r, \qquad (10)$$

that we can rewrite as

$$\left[\mathbf{B}_r\right]_{l,m\,(v,s)} = W_N^{[\![lmN/r + \lfloor v/r^s \rfloor l\, r^s]\!]_N} \qquad (11)$$

As a result, the operation of a radix-r for the DIF FFT is formulated by, the column vector:

$$\mathbf{X}_{(u,v,s)} = \mathbf{B}_{r\;DIF}\mathbf{x},$$

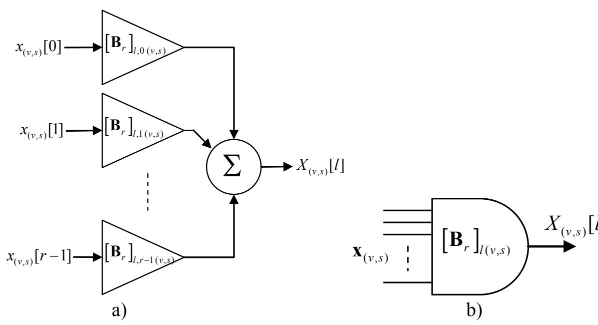
where the lth output is

$$X_{(v,s)}[l] = \sum_{m=0}^{r\text{-}1} x_{(v,s)}[m] W_N^{[\![lmN/r + \lfloor k/r^s \rfloor l\, r^s]\!]_N}. \qquad (12)$$

With the same reasoning as above, the operation of a radix-r DIT FFT can be derived.

The conceptual key to the modified radix-r FFT butterfly is the formulation of the radix-r as composed butterflies with identical structures and a systematic means of accessing the corresponding multiplier coefficients. This enables the design of processing element (PE), called Butterfly PE (BPE) shown in Fig. 1, to maximize the data throughput, we can utilize r or r-1 complex multipliers in parallel to implement each of the radix-r butterfly computations.

Fig. 2 shows the radix-r BPE for one element output l. For a single processor environment, this type of BPE would result in decrease in time delay for the complete FFT by a factor of $O(r)$. The modified radix-r FFT butterfly based on the proposed BPEs presents a useful aspect by applying parallel multiprocessing environments as shown in Fig. 3 [3].



$x_{(v,s)}[0]$
$[\mathbf{B}_r]_{l,0\,(v,s)}$
$x_{(v,s)}[1]$
$[\mathbf{B}_r]_{l,1\,(v,s)}$
$x_{(v,s)}[r-1]$
$[\mathbf{B}_r]_{l,r-1\,(v,s)}$
Σ
$X_{(v,s)}[l]$
$\mathbf{x}_{(v,s)}$
$[\mathbf{B}_r]_{l\,(v,s)}$
$X_{(v,s)}[l]$
a)
b)

Fig. 1 BPE for FFT Radix-r (a) using $\mathbf{B}_r$ in (11) and the symbol (b).

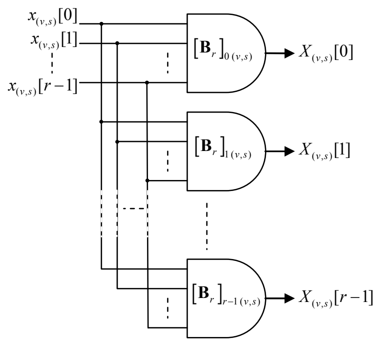


Fig. 2 Maximize the data throughput using r BPEs in parallel.

3. DFT Factorization

For a given $r \times r$ square matrix $\mathbf{T}_r$ and for a given column vector $\mathbf{x}[n]$ of size N, we define a special product expressed with the operator $\hat{*}_{(\alpha,\gamma,\beta)}$ (product of radix-α performed on γ column vector of size β) by the following operation, where the γ column vectors are subsets of $\mathbf{x}[n]$ picked up at a stride α

$$X[k] = \hat{*}_{(r,r,N/r)}\left(\mathbf{T}_r, \begin{bmatrix} x[rn] \\ x[rn+1] \\ \vdots \\ x[rn+(r-1)] \end{bmatrix}\right) = \mathbf{T}_r \begin{bmatrix} x[rn] \\ x[rn+1] \\ \vdots \\ x[rn+(r-1)] \end{bmatrix} \qquad (13)$$

$$X[k] = \begin{bmatrix} T_{0,0} & T_{0,1} & \cdots & T_{0,r-1} \\ T_{1,0} & T_{1,1} & \cdots & T_{1,r-1} \\ \vdots & \vdots & \ddots & \vdots \\ T_{r-1,0} & T_{r-1,1} & \cdots & T_{r-1,r-1} \end{bmatrix} col\left(x[rn+j_0]\right)_{j_0=0,1,\ldots,r-1} \qquad (14)$$

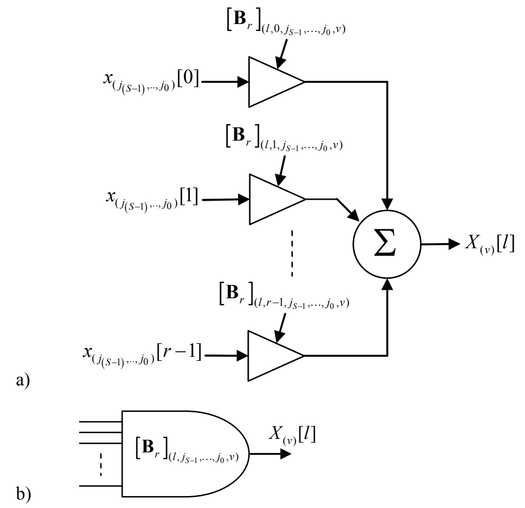
$$X_v[l] = \left[\sum_{j_0=0}^{r-1} \left[\mathbf{T}\right]_{l,j_0} x[rn+j_0]\right] \qquad (15)$$

for $v = 0,1,\ldots,N/r-1$ and j_0=0,1,…,r-1; $\mathbf{X} = \left[X[0], X[1], \cdots, X[(N/r)-1]\right]^T$ is a column vector. We can generalized to a r column vectors of length $\lambda\beta$ where λ is a power of r in which the l^{th} element $X[l]$ of the v^{th} product $X_{(v)}[l]$ is labeled as

$$l_{(v)} = j_0\lambda\beta + v \qquad (16)$$

for v=0,1,...,$\lambda\beta$-1. Special properties are shown in Appendix.

Based on our proposition in the previous section, Eq. (1) for the first factorization may be rewritten as



$[\mathbf{B}_r]_{(l,0,j_{S-1},\ldots,j_0,v)}$
$x_{(j_{(S-1)},\ldots,j_0)}[0]$
$[\mathbf{B}_r]_{(l,1,j_{S-1},\ldots,j_0,v)}$
$x_{(j_{(S-1)},\ldots,j_0)}[1]$
Σ
$X_{(v)}[l]$
$[\mathbf{B}_r]_{(l,r-1,j_{S-1},\ldots,j_0,v)}$
$x_{(j_{(S-1)},\ldots,j_0)}[r-1]$
a)
$[\mathbf{B}_r]_{(l,j_{S-1},\ldots,j_0,v)}$
$X_{(v)}[l]$
b)

Fig. 3 Radix-*r* OSPE (a) using $\mathbf{B}_{r\,DIF}$ in (11) and the symbol (b).

$$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{kn}=\widehat{*}_{(r,r,N/r)}\left(T_r,\begin{bmatrix}\sum\limits_{n=0}^{(N/r)-1}x[rn]W_N^{rnv_0}\\ \sum\limits_{n=0}^{(N/r)-1}x[rn+1]W_N^{(rn+1)v_0}\\ \vdots\\ \sum\limits_{n=0}^{(N/r)-1}x[rn+(r-1)]W_N^{(rn+(r-1))v_0}\end{bmatrix}\right) \tag{17}$$

for v_0=0,1,..., (N/r)-1, and n=0,1,...,N-1. Since

$$W_N^{rnk}=W_{N/r}^{nk} \tag{18}$$

the Eq. (17) becomes

$$X[k]=\widehat{*}_{(r,r,N/r)}\left(T_r,\begin{bmatrix}\sum\limits_{n=0}^{(N/r)-1}x[rn]\,w_{N/r}^{nv_0}\\ W_N^{v_0}\sum\limits_{n=0}^{(N/r)-1}x[rn+1]W_{N/r}^{nv_0}\\ \vdots\\ W_N^{(r-1)kv}\sum\limits_{n=0}^{(N/r)-1}x[rn+r-1]W_{N/r}^{nv_0}\end{bmatrix}\right) \tag{19}$$

which for simplicity may be expressed as

$$X[k]=\widehat{*}_{(r,r,N/r)}\left(\mathbf{T}_r\left[W_N^{j_0v_1}\right],col\left(\sum_{n=0}^{(N/r)-1}x[rn+j_0]W_{N/r}^{nv_0}\right)\right) \tag{20}$$

where for simplification in notation the column vector in (20) is set equal to:

$$\begin{bmatrix}\sum\limits_{n=0}^{(N/r)-1}x[rn]w_{N/r}^{nv_0}\\ w_N^{v_0}\sum\limits_{n=0}^{(N/r)-1}x[rn+1]w_{N/r}^{nv_0}\\ \vdots\\ w_N^{(r-1)v_0}\sum\limits_{n=0}^{(N/r)-1}x[rn+r-1]w_{N/r}^{nv_0}\end{bmatrix}=col\left(\sum_{n=0}^{(N/r)-1}x[rn+j_0]W_{N/r}^{nv_0}\right) \tag{21}$$

for j_0=0,1,…,r-1, v_0=0,1,…,(N/r)-1 and $\left[\mathbf{W}_N^{j_0v_0}\right]=\mathrm{diag}\left(W_N^0,W_N^{v_0},\cdots,W_N^{(r-1)v_0}\right)$.

For the second factorization, (31) is factored as follow:

$$X[k]=\widehat{*}_{\left(r,r,\frac{N}{r}\right)}\left(\mathbf{T}_r\left[W_N^{j_0v_0}\right],\begin{bmatrix}\widehat{*}_{\left(r,r^2,\frac{N}{r^2}\right)}\left(\mathbf{T}_r,\begin{bmatrix}\sum\limits_{n=0}^{\left(\frac{N}{r^2}\right)-1}x[r(rn)]W_{N/r^2}^{nv_1}\\ \vdots\\ W_N^{r(r-1)v_1}\sum\limits_{n=0}^{\left(\frac{N}{r^2}\right)-1}x[r(rn+(r-1))]W_{N/r^2}^{nv_1}\end{bmatrix}\right)\\ \widehat{*}_{\left(r,r^2,\frac{N}{r^2}\right)}\left(\mathbf{T}_r,\begin{bmatrix}\sum\limits_{n=0}^{\left(\frac{N}{r^2}\right)-1}x[r(rn)+1]W_{N/r^2}^{nv_1}\\ \vdots\\ W_N^{r(r-1)v_1}\sum\limits_{n=0}^{\left(\frac{N}{r^2}\right)-1}x[(r(rn+(r-1))+1)]W_{N/r^2}^{nv_1}\end{bmatrix}\right)\\ \vdots\\ \widehat{*}_{\left(r,r^2,\frac{N}{r^2}\right)}\left(\mathbf{T}_r,\begin{bmatrix}\sum\limits_{n=0}^{\left(\frac{N}{r^2}\right)-1}x[r(rn)+(r-1)]W_{N/r^2}^{nv_1}\\ \vdots\\ W_N^{r(r-1)v_1}\sum\limits_{n=0}^{\left(\frac{N}{r^2}\right)-1}x[r(rn+r-1)+(r-1)]W_{N/r^2}^{nk_1}\end{bmatrix}\right)\end{bmatrix}\right) \tag{22}$$

which could be simplified as equation (23), for j_0=0,...,r-1, j_1=0,...,r-1, v_1=0,1,…,N/r^2-1, and $\left[\mathbf{W}_N^{rj_1v_1}\right]=\mathrm{diag}\left(W_N^0,W_N^{rv_1},\cdots,W_N^{r(r-1)v_1}\right)$. If the factorization process will continue till we get r^s transform of size r, then Eq. (1) will be expressed as Eq. (24) and $\left[\mathbf{W}_N^{r^{(s)}j_sv_s}\right]=\mathrm{diag}\left(W_N^0,W_N^{r^sv_s},\cdots,W_N^{r^s(r-1)v_s}\right)$.

In DSP Layman language, the factorization of an FFT can be interpreted as dataflow diagram (or Signal Flow Graph), which depicts the arithmetic operations and their dependencies. To be noted that the dataflow diagram is read from left to right we will obtain the decimation in frequency algorithm and where λ in Eq. (16) is equal to $r^{(-1)}$, meanwhile if the dataflow diagram is read from right to left we will obtain the decimation in time algorithm and where λ in (16) is equal to r.

4. The One Stage FFT

Eq. (30) could be developed according to our special product,

$$X[k]=\widehat{*}_{(r,r,N/r)}\left(\mathbf{T}_r\left[W_N^{j_0v_0}\right],col\left(\widehat{*}_{(r,r^2,N/r^2)}\left(T_r\left[W_N^{rj_1v_1}\right],col\left(\sum_{n=0}^{v_1-1}x[r^2n+rj_1+j_0]w_{N/r^2}^{nv_1}\right)\right)\right)\right) \tag{23}$$

$$X[k]=\widehat{*}_{\left(r,r^s,k_s\right)_{s=0,1,\ldots,\log_r N-2}}\left(\mathbf{T}_r\left[W_N^{r^sj_sv_s}\right],col\left(\sum_{n=0}^{v_s-1}x[r^{(s+1)}n+r^{(s)}j_s+\ldots+j_0]W_{N/r^{s+1}}^{nv_s}\right)\Bigg|_{v_s=0,1,\ldots,N/r^{s+1}-1}\right) \tag{24}$$

Table I Number of cycles need to execute a 4096-points FFT for different radices by factoring the adder matrix.

Cycles Requirements	Radix – 2	Radix – 4	Radix – 8
Phases	2	4	7
Memory accesses	71680	17408	5632
Complex multiplication	24576	5120	3072
Complex addition	24576	12288	6144

Table II Number of cycles needs to execute a 4096-points FFT for different radices by implementing *r* BPEs in parallel (Fig. 2).

Cycles Requirements	Radix – 8	Radix – 16
Phases	4	3
Memory accesses	3072	1536
Complex multiplication	1536	512
Complex addition	2048	768

Table III Number of cycles needs to execute a 4096-points FFT for different radices by using *r* OSPE (Fig. 4).

Cycles Requirements	Radix – 8	Radix – 16
Phases	1	1
Memory accesses	1024	512
Complex multiplication	512	16
Complex addition	512	16

$\hat{*}_{(\alpha,\gamma,\beta)}$, and knowing that

$$\mathbf{T}_r = \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^{N/r} & W_N^{2N/r} & \cdots & W_N^{(r-1)N/r} \\ W_N^0 & W_N^{2N/r} & W_N^{4N/r} & \cdots & W_N^{2(r-1)N/r} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{(r-1)N/r} & W_N^{2(r-1)N/r} & \cdots & W_N^{(r-1)^2 N/r} \end{bmatrix} \qquad (25)$$

where $[\mathbf{T}_r]_{l,m} = W^{\left[\!\left[lm\frac{N}{r} \right]\!\right]_N}$ (26), *l*=0,...,*r*-1, and *m*=0,...,*r*-1, therefore, (21) could be simplified as:

$$X_v[l] = \sum_{j_0=0}^{r-1} \sum_{j_1=0}^{r-1} \cdots \sum_{j_S=0}^{r-1} \sum_{n=0}^{r-1} x[r^S n + r^{S-1} j_{S-1} + \cdots + j_0] W_N^{\left[\!\left[\frac{JNl}{r} + \left(J + \frac{Nn}{r}\right) v \right]\!\right]_N} \qquad (27)$$

where $J = r^{S-1} J_{S-1} + r^{S-2} j_{S-2} + \cdots + r j_1 + j_0$ and for $j_s = 0,1,\ldots,r-1,\ s \in [0, S-1]$, *l*=0,1,…,*r*-1, *v*=0,1,...,(*N*/*r*)-1, *S*=*log*$_r$*N*. The *l*th output of X is stored at the address memory location given by the write address generator (WAG):

$$WAG = l(N/r) + v\,. \qquad (28)$$

Finally, we can represent the execution of FFT in one stage (or phase), by adopting the following notations:

$$x_{(j_{(S-1)},\ldots,j_0)}[m] = x[r^S m + r^{S-1} j_{S-1} + \cdots + j_0] \qquad (29)$$

$$[\mathbf{B}_r]_{(l,m,j_s\ldots,j_0,v)} = W_N^{\left[\!\left[\frac{JNl}{r} + \left(J + \frac{Nm}{r}\right) v \right]\!\right]_N}. \qquad (30)$$

Fig. 3 illustrates the radix-*r* one stage processing element (OSPE) in which *r* multipliers are implemented in parallel and executed in one cycle. To increase the data throughput, we can easily increase the degree of parallelism as shown in Fig. 4. All FFT results from Fig. 4 are obtained in one clock cycle to satisfy high sustained throughput applications. The data, $[\mathbf{B}_r]_{(l,m,j_s\ldots,j_0,v)}$, is localized at each multiplier and use an address generator based on a simple digital counters.

The one stage FFT structure is suitable to customize the hardware implementation take into account the VLSI constraints such as data throughput, area, and power consumption. The localized communication, regularity and recursiveness of equations make the flexibility to satisfy a large application domains. The proposed structure gives us the ability to divide a process into serial and parallel portions (or pure parallel portions) where the parallel parts are executed concurrently.

Tables I to III show the evaluation for adder matrix [4] and proposed FFT (Fig. 2 and 4). Table II shows more clock cycles than Table 1 but with a lower number of hardware resources by applying a time multiplex implementation of the BPE (Fig. 2). However, we drastically reduce the clock cycles using *r* OSPE in parallel (Fig. 4) as shown in Table III.

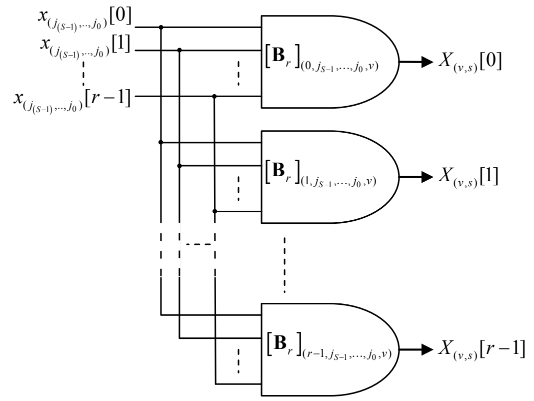

$x_{(j_{(S-1)},\ldots,j_0)}[0]$
$x_{(j_{(S-1)},\ldots,j_0)}[1]$
$x_{(j_{(S-1)},\ldots,j_0)}[r-1]$
$[\mathbf{B}_r]_{(0,j_{S-1},\ldots,j_0,v)}$
$[\mathbf{B}_r]_{(1,j_{S-1},\ldots,j_0,v)}$
$[\mathbf{B}_r]_{(r-1,j_{S-1},\ldots,j_0,v)}$
$X_{(v,s)}[0]$
$X_{(v,s)}[1]$
$X_{(v,s)}[r-1]$

Fig. 4 Maximize the data throughput using *r* OSPE in parallel.

5. Conclusion

Finally this paper has presented an efficient way of implementing the FFT process by mean of the one iteration radix-*r* kernel where a serial parallel model and a pure parallel model have been represented. Also, it has been argued that a reduction in the chip size, a reduction of its power consumption and an increase of the performance of the system could be achieved.

References

[1] J.W. Cooley and J.W. Tukey, "An algorithm for the machine calculation of complex fourier series", *Math. Comput.*, 19, pp. 297-301, April 1965.

[2] T. Widhe, J. Melander, and L. Wanhammar, "Design of efficient radix-8 butterfly PEs for VLSI", *IEEE Int. Symposium on Circuit and Systems*, vol. 3, pp. 2084-2087, June 1997.

[3] M. Jaber, "Butterfly Processing Element for Efficient Fast Fourier Transform Method and Apparatus", US Patent No. 6,751,643, 2004.

[4] Y. Jung, H. Yoon and J. Kim, "New Efficient FFT Algorithm and Pipeline Implementation Results for OFDM/DMT Applications", *IEEE Trans. on Consumer Electronics,* vol. 49, no. 1, pp. 14-20, Feb. 2003.

Appendix

Properties of special product $\hat{*}_{(r,r,\beta)}$

Lemma

$$\begin{aligned} X[k] &= \hat{*}_{(r,r,\beta)}\left(\mathbf{T}_r,\left(\mathbf{W}_r col\left[x[rn + j_0]\right]\right)\right) \\ &= \hat{*}_{(r,r,\beta)}\left(\mathbf{T}_r\mathbf{W}_r,\left(col\left[x[rn + j_0]\right]\right)\right) \end{aligned} \qquad (31)$$

Proof:

$$X[k] = \hat{*}_{(r,r,\beta)}\left(\mathbf{T}_r,\left(W_r col\left(x[rn + j_0]\right)\right)\right) = \mathbf{T}_r\left(W_r col\left[x_{(rn+j_0)}\right]\right)$$

$$X[k] = \left(\mathbf{T}_r W_r\right) col\left[x_{(rn+j_0)}\right] = \hat{*}_{(r,r,\beta)}\left(\left(\mathbf{T}_r W_r\right),\left(col\left[x_{(rn+j_0)}\right]\right)\right)$$