Optimal Circuits for Parallel Bit Reversal

Ren Chen and Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles, USA 90089
{renchen, prasanna}@usc.edu

ABSTRACT

In this paper, we develop novel parallel circuit designs for calculating the bit reversal. To perform bit reversal on 2^n data words, the designs take 2^k ($k < n$) words as input each cycle. The circuits consist of concatenated single-port buffers and 2-to-1 multiplexers and use minimum number of registers for control. The designs consume minimum number of single-port memory banks that are necessary for calculating continuous-flow bit reversal, as well as near optimal 2^n memory words. The proposed parallel circuits can be built for any given fixed k and n , and achieve superior performance over state-of-the-art for calculating the bit reversal in parallel multi-path FFT architectures.

CCS Concepts

• **Hardware** → **Digital signal processing; Application specific integrated circuits; Sequential circuits;**

1. INTRODUCTION

Bit reversal has been widely used for calculating the output frequencies of the Fast Fourier Transforms (FFT)s [13]. Given a set of data, a bit reversal reorders the indexed data according to a reversing of the bits of the data index [16]. As all the inputs are available concurrently, a straightforward bit reversal implementation can be reordered hardware wires. However, a compact pipelined design is more desirable for processing a long data sequence, considering the routing complexity and I/O bandwidth limitation.

Bit reversal has been extensively studied for decades in various research areas [21, 22, 13, 15, 10]. Many existed approaches in the literature focus on efficient algorithm theories to compute the bit reversal of a data sequence stored in a memory [21, 22]. These approaches are highly suitable for microprocessor architectures. Efficient bit reversal design is particularly important in the FFT hardware implementations [13, 15]. Based on the type of data storage used, previous work on the circuit design for bit reversal can be classified into memory-based designs and register-based designs. According to [6, 17], delay feedback/commutator is proposed to perform bit reversal in the traditional VLSI implementation for FFT. Input samples are bit reversed using register-based delay feedback/commutator in these folded FFT architectures to achieve high computational performance per

unit area. In this context, realizing the bit reversal becomes a different problem from reordering data stored in a memory. With the aim of minimizing resource consumption, register-based serial designs for bit reversal are developed in [14] and further adopted in the implementation of radix feedforward FFT architectures [15]. To improve the throughput, parallel bit reversal circuit designs are developed in [11, 12] such that input samples can be processed by multi-path delay feedback (MDF) and multi-path delay commutator (MDC) FFT architectures.

In addition to the above designs, other relevant research works are proposed to realize a specific family of data permutations including stride permutation and bit reversal. In [7, 9], parallel streaming permutation structures are proposed to obtain high performance hardware implementations for the FFT. In [18], the authors present a register-based permutation network for stride permutations for array processor. Bit reversal is realized by a combination of different stride permutations. The proposed network supports any stride of power-of-two and achieves high resource efficiency. In [20], a memory-based permutation approach has been proposed; the permutation network is proved mathematically able to realize any given bit-index permutations including bit reversal using dual-ported memories. These previous works are more generalized approaches without optimization specifically for bit reversal.

In this paper, we address the problem of calculating bit reversal on parallel continuous data flows and propose an optimal solution to this problem. The solution consists of circuits which only require single-port buffers, 2-to-1 multiplexers, as well as simple control circuits. The proposed circuits are proven to be optimal in two senses. At first, they achieve the lower bound of the number of memory banks with near optimal memory efficiency, i.e., the continuous-flow bit reversal cannot be realized with fewer storage banks. Secondly, the circuits achieve optimal control mechanism with regarding to the number of control registers. Specifically, our main contributions are the following:

- Parallel circuits for calculating bit reversal on parallel data inputs (Section 4).
- Parameterized bit reversal designs with respect to input size and data parallelism. (Section 4.2).
- Demonstrated optimal designs achieving the lower bound of the number of memory banks as well as near optimal memory efficiency. (Section 4.3).
- Optimal control mechanism for supporting reordering outputs of power-of-2 FFTs (Section 4.2).

The rest of the paper is organized as follows. Section 2 introduces the background. Section 3 defines the parallel bit reversal problem. Section 4 presents our proposed parallel

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DAC '17, June 18-22, 2017, Austin, TX, USA

© 2017 ACM. ISBN 978-1-4503-4927-7/17/06...\$15.00

DOI: <http://dx.doi.org/10.1145/3061639.3062295>

Figure 1: Data flow in the parallel- 2^k bit reversal design for input size 2^n , $d = 2^k - 1$ and $s = 2^{n-k} - 1$.

bit reversal circuit designs. Section 5 presents experimental results. Section 6 concludes the paper.

2. BACKGROUND

Bit reversal: As an important permutation pattern, bit reversal has been extensively used in FFTs [6]. Bit reversal can be mathematically represented using a two-dimensional bit matrix B_N such that $y = B_N \cdot x$. B_N maps each element $x[i]$ in x to the position given by reversing the $\log_2 N$ -bit binary representation of the index i . Assuming $y[j] = x[i]$ and the binary representation of i is $b_0 b_1 \dots b_{\log N - 1}$, then:

$$j = \sigma(b_0 b_1 \dots b_{\log N - 1}) = b_{\log N - 1} \dots b_1 b_0 \quad (1)$$

where $\sigma()$ is the bit reversing operation. Bit reversal permutation matrix is also known as the base- t digital reversal permutation when $t = 2$. The permutation matrix of the base- t digital reversal permutation is denoted as $B_{N,t}$:

$$B_{N,t} = \prod_{i=1}^{\log_t N} (I_{N/\log_t^i N} \otimes P_{\log_t^i N, t}) \quad (2)$$

Where I is the identity matrix, $P_{\log_t^i N, t}$ is the stride permutation introduced next, and $\otimes$ is the well-known matrix operation *tensor product* (Kronecker product) [19]. For example, $B_{4,2}$ performs $x_0, x_1, x_2, x_3 \rightarrow x_0, x_2, x_1, x_3$ and $y = B_{4,2} \cdot x$ can be represented as

$$\begin{pmatrix} y_0 & y_1 & y_2 & y_3 \end{pmatrix}^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \end{pmatrix}^T$$

Stride Permutation: Another class of important data permutations in FFT algorithms are stride permutations. Given an m -element data vector x and a stride t ($1 \leq t \leq m - 1$), the data vector y produced by the stride-by- t permutation over x is given as $y = P_{m,t} x$, where $P_{m,t}$ is a permutation matrix. $P_{m,t}$ is an $m \times m$ bit matrix such that

$$P_{m,t}[i][j] = \begin{cases} 1 & \text{if } j = (t \times i) \bmod m + (\lfloor t \times i / m \rfloor) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where mod is the modulus operation and $\lfloor \cdot \rfloor$ is the floor function.

3. PROBLEM DEFINITION

Bit reversal on serial data has been well studied [6, 14, 17]. Recently hardware designs for parallel bit reversal are proposed for high throughput MDC or MDF FFT architectures [11, 12]. Fig. 1 shows the data flow to realize bit reversal on a data sequence of size 2^n , each time taking 2^k data elements as the inputs. In such case, as the data

Figure 2: Generating parallel-2 bit reversal design for input size 2^n (A_0 and A_1 are data buffer addresses)

inputs cannot be all available currently, designing a specialized hardware block for performing bit reversal on parallel data becomes challenging as data elements need to be moved across the temporal bound, i.e., a data element previously input in input cycle i needs to be output in output cycle j , where $i \neq j$. This feature requires register files or storage elements such as memory banks to be employed so that a data element can be output after a specific amount of delay.

To formulate this problem, we define the **parallel- 2^k bit reversal** problem as: to design a parallel hardware design with an available data parallelism of p ($p = 2^k$). p^1 is defined as the number of parallel inputs processed per cycle. Therefore, to calculate the bit reversal of a 2^n -element data sequence, the inputs enter the hardware design over 2^{n-k} ($k < n$) consecutive cycles. After a specific amount of delay t , the sequence is reordered as specified by the bit reversal and output over 2^{n-k} consecutive cycles. Such design can be pipelined to process continuous data flows, thus the peak throughput is p output elements per cycle and the latency is t .

4. DESIGN APPROACH

4.1 Parallel-2 Bit Reversal

Parallel- 2^k bit reversal can be formulated as a permutation matrix problem [20] or a graph problem. In this paper, we propose a mapping methodology translating this challenging problem into a classic switch network routing problem. Fig. 2 illustrates the key idea of the proposed mapping methodology, where the upper switch network is the classic Benes network [5] which can realize bit reversal between the input and the output by configuring its 2-to-2 switches. The first step to route the Benes network is to decompose it into two stages of switches and two middle sub-networks. Fig. 2 shows that each 2×2 input/output switch has exactly one link connecting to the upper subnetwork or the lower sub-network. The input and output switch stages need to be configured such that the input data are routed to the right

¹Notations including p , 2^k , N and 2^n ($p = 2^k$, $N = 2^n$) will be used throughout this paper.

Figure 3: 2-to-2 switch in different states

Figure 4: Functional structure of the employed data buffer and its behavior in the read-before-write mode

location at the output based on the mapping of bit reversal. The switch configuration bits can be easily computed by the classic looping algorithm [5]. The two subnetworks can be further decomposed and routed in a recursive manner such that $2 \log N + 1$ stages of switches are used and configured to realize a specific routing when the input size is N . We first propose a parallel-2 bit reversal design by vertically folding the topology of the upper network in Fig. 2. In Fig. 2, the resulting datapath of our design has a three-stage structure, including two stages of switches and one memory (buffer) stage. The 2×2 switch in the input/output stage is run time reconfigured to realize the connection of switches at the input/output stage of Benes network. The upper (lower) subnetwork in the middle stage of Benes network is replaced with a memory bank in the datapath as shown in Fig. 2, where this substitution relationship is emphasized using colored wire connections. The mapping between the input and the output of a subnetwork determines the number of cycles to be buffered for each data element in a corresponding memory bank.

The proposed parallel-2 design shown in Fig. 2 is parameterizable with respect to input size 2^n . It only requires two 2-to-2 switches, each having two 2-to-1 multiplexers and two single-port data buffers, each of size 2^{n-1} . Fig. 4 shows the data buffer structure which has one address port, one data port for read and one data port for write, where the two data ports share the single address port. To realize the bit reversal, our design requires the data buffer to be accessed in the read-before-write mode, i.e., given a buffer address, the old data previously stored at the write address appears first on the output latches, while the input data is being stored in the buffer. The access behavior of the buffer in the read-before-write mode is shown in Fig. 4. Fig. 3 shows the circuit behaviors of the 2-to-2 switch in different states specified by the value of the control bit. When the control bit value is zero, the 2-to-2 switch bypass the inputs, otherwise exchanges the inputs and routes them to the output. Such a switch can be implemented using two 2-to-1 multiplexers.

Table 1 presents the values of the control bits including cl , A_0 , A_1 , cr shown in Fig. 2 for computing bit reversal. Input

Table 1: Control bit values for continuous-flow bit reversal when $2^n = 8$

Time (cycles)	Values of control bits			
	cl	A_0	A_1	cr
0	0	0	0	0
1	0	1	1	0
2	1	2	2	1
3	1	3	3	1
4	0	0	2	0
5	0	1	3	0
6	1	2	0	1
7	1	3	1	1

Figure 5: Data flows in parallel-2 bit reversal for $2^n = 8$

data are fed into the circuits starting from cycle 0. For the first two cycles, the inputs bypass the 2-to-2 switch as $cl = 0$. For the next two cycles, the two inputs are swapped and routed to the output as $cl = 1$. During cycles 0, 1, 2 and 3, the inputs are first written into the data buffers, then, in the subsequent four cycles, data stored in buffers are read out and routed by the output switch. The complete data flow in the parallel-2 design could be obtained using Table 1. For simplicity, we only show the data flow during the input cycles and the output cycles in Fig. 5. The control bit values will be repeatedly updated based on Table 1 when processing continuous data flows. Fig. 4 illustrates how sub data vectors x , y , z which belong to different data flows are temporally reordered in the same pattern, where v_0, v_1, v_2, v_3 ($v = x$ or y or z) are buffered for 6, 6, 2, 2 cycles, respectively.

4.2 Parallel- 2^k Bit Reversal

Fig. 6(a) illustrates our proposed designs for parallel- 2^k bit reversal. Similarly, the 2^n inputs are divided into several 2^k -element sub-vectors, which are fed into the parallel- 2^k bit reversal design over 2^{n-k} consecutive cycles. Therefore, data parallelism is 2^k . During the output phase, the design produces 2^k outputs per cycle. The notations used in Fig. 6 are illustrated as below:

- L_{ij} ($0 \leq i \leq s, 0 \leq j \leq p-1$): a 2-to-2 switch (see Fig. 3) at the input stage i of the design.
- R_{ij} ($0 \leq i \leq s, 0 \leq j \leq p-1$): a 2-to-2 switch at the output stage i of the design.
- W_i/W'_i : 2^k -to- 2^k fixed wire interconnection, $i = 0, 1, \dots, s$.
- M_i : single-port memory banks (see Fig. 4) at the middle stage of the design.

The proposed design shown in Fig. 6 is obtained by extending the mapping approach introduced in Section 4.1, such that the Benes network is recursively decomposed into 2^k subnetworks and further vertically folded motivated by the idea of time multiplexing. Such design is parameterizable with respect to input size 2^n and data parallelism 2^k . Based on such mapping methodology, the theorem below can be obtained:

Figure 6: (a) Circuit design for parallel- 2^k bit reversal: for each L_{ij} and R_{ij} , i and j denote stage index and vertical index, respectively. (b) Parallel-4 bit reversal design for input size 2^n

Figure 7: Sub-sections in Memory Address A_j

THEOREM 4.1. *Parallel- 2^k bit reversal $B_{2^n,2}$ can be performed by the proposed circuit design when W_i realizes $P = I_{2^i} \otimes P_{p/2^i,2}$ and W'_i realizes $P = I_{2^i} \otimes P_{p/2^i,p/2^{i+1}}$ ($0 \leq i \leq \log_2 p - 1$).*

Proof to Theorem 4.1 can be found in [8]. Therefore, the parallel-4 bit reversal design is obtained as shown in Fig. 6(b). The control bits for the design in Fig. 6(a) are denoted as below:

- cl_i ($0 \leq i \leq s$): one bit control determining the state of the switches including $L_{i0}, L_{i1}, \dots, L_{id}$. Its value is either 0 or 1. Note that the $d + 1$ switches at stage i share cl_i .
- A_j ($0 \leq j \leq p - 1$): $n-k$ bits used as the address for data buffer M_j .
- cr_i ($0 \leq i \leq s$): one bit control determining the state of the switches including $R_{i0}, R_{i1}, \dots, R_{id}$. Its value is either 0 or 1.

Table 2: Control bit values for parallel- 2^k bit reversal and $N = 2^n$

t (cycle number)	Values of control bits				
	cl_i	A_j			cr_i
		$A_j[0]$	$A_j[1]$	$A_j[2]$	
0	$f(i, t)$	0			$f(i, t)$
1		1			
...		...			
$2^{n-k}-1$		$2^{n-k}-1$			
2^{n-k}					
$2^{n-k}+1$		$g_0(j, t)$	$g_1(j, t)$	$g_2(j, t)$	
...					
$2^{n-k+1}-1$					

Note that the control bits are not shown in Fig. 6 for simplicity. Table 2 shows the values of the control bits for calculating parallel- 2^k bit reversal. All the control bit values are computed using the Benes network based mapping ${}^2I_{2^i}$ is the identity matrix and $\otimes$ is the tensor (or Kronecker) product [19].

approach (see Section 4.1). These values are further represented using functions including $f(i, t)$, $g_0(j, t)$, $g_1(t)$, and $g_2(t)$. Values of cl_i or cr_i are determined by function $f(i, t)$, where i is the stage index, and t is the cycle number. $f(i, t)$ is defined as:

$$f(i, t) = \begin{cases} 0 & \text{if } (t \bmod 2^{n-k-i}) < 2^{n-k-1-i} \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

where $f(i, t)$ switch between 0 and 1 periodically. Based on $f(i, t)$, control bit values of all cl_i and cr_i can be obtained using a $(n - k)$ -bit counter.

Fig. 7 shows that each address A_j ($0 \leq j \leq d$) has $n - k$ bits, which are divided into three sections including $A_j[0]$, $A_j[1]$, and $A_j[2]$, determined by $g_0(j, t)$, $g_1(t)$, $g_2(t)$, respectively. Note that values of $A_j[1]$ or $A_j[2]$ at cycle t are the same for all j ($0 \leq j \leq d$). When $k \geq n - k$, $A_j[0] = A_j$. Otherwise, $A_j[0]$ is obtained by the most significant l_0 ($l_0 = k$) bits in A_j , $A_j[1]$ consists of l_1 ($l_1 = (n - 2k) \bmod 4$) bits, and $A_j[1]$ includes the least significant l_2 ($l_2 = n - 2k - l_1$) bits in A_j . Furthermore, $g_0(j, t)$ is defined as:

$$g_0(j, t) = a_0 \oplus b_j \quad (5)$$

where a_0 is the binary representation of the result by $(t/c_0) \bmod 2^k$, and c_0 is a power-of-two constant. b_j is the binary representation of j . The values computed by $g_0(j, t)$ can be dynamically produced by the control unit in Fig. 6(a) using a $(k + \log_2 c_0)$ -bit counter and ignoring its least significant $\log_2 c_0$ bits. Then, $g_1(j, t)$ is defined as:

$$g_1(t) = \sigma(a_1) \quad (6)$$

where $\sigma()$ is the bit reversing operation defined in Section 2, a_1 is the binary representation of the result by $(t/c_1) \bmod 2^{l_1}$, and c_1 is a power-of-two constant. In the control unit, to update values of $A_j[1]$ determined by $g_1(t)$, a $(l_1 + \log_2 c_1)$ -bit counter ignoring its least significant $\log_2 c_1$ bits is satisfied. The bit reversing operation can be easily realized by permuted wires in hardware. $g_2(t)$ is defined as:

$$g_2(t) = \{\sigma(a_{20}), \sigma(a_{21}), \dots, \sigma(a_{2m})\} \quad (7)$$

where $\{\}$ is the concatenation operation for bit representations [7], and $m = l_2/4 - 1$. a_{2q} ($0 \leq q \leq m$) is the binary representation of the result by $(t/c_{2q}) \bmod 16$, and c_{2q} is a power-of-two constant. In the control unit, bit values of a $(l_2 + \log_2 c_{2q})$ -bit counter ignoring its least significant $\log_2 c_{2q}$ -bit can be employed for updating $A_j[2]$.

In Table 2, $A_j = t$ when $0 \leq t \leq 2^{n-k}-1$, and then A_j is

Table 3: Comparison of several bit reversal circuit designs

	Supported FFT architecture	Supported data parallelism	# of data buffers	Each buffer port type	Total buffer size (words)	Size to support continuous flow	# of mux ⁽¹⁾	Throughput	Latency
[14]	SDF	Serial data	2	two-port	$(\sqrt{N}-1)^2$	$(\sqrt{N}-1)^2$	Not shown	1	$(\sqrt{N}-1)^2$
[4]	Real-valued FFT	Serial data	1	two-port	$(5N/8)-3$	$(5N/8)-3$	Not shown	1	$(5N/8)-3$
[23]	MDC	Parallel (8)	8	two-port	$p \times N$	$p \times N$	Not shown	p	Not shown
[11]	MDC/MDF	Parallel (2/4/8)	$2p$	single-port	N	N	$2p^2-p$	p	$(N/p)-Y^{(2)}$
[12]	MDC	$2^k (2^k < \sqrt{N})$	$X^{(4)}/2 + 3p/2$	two-port	$N - C^{(3)}\sqrt{N} + p$	$N - C^{(3)}\sqrt{N} + p$	$X^{(4)}$	p	$N/p - C^{(3)}\sqrt{N}/p + 1$
[20]	MDC	2^k	p	two-port	N	$2N$	$\leq 2p \log p$	p	N/p
This work	MDC	2^k	p	single-port	N	N	$2p \log p$	p	N/p

¹ 2-to-1 multiplexer, ² $Y = 2^a - 1$, $a = (\log_2(N/2p^2))/2$ or $(\log_2(N/2p^2))/2 + 1$, ³ C is a constant and $C \leq 2$, ⁴ $X = 2p \log_2(N/p^2)/2 + 4p^2 - 4p$, and $X > 2p \log_2 N$

determined by $g_0(j, t)$, $g_1(t)$, $g_2(t)$, where c_0, c_1 and c_{2q} are pre-computed, during cycle t ($2^{n-k} \leq t \leq 2^{n-k+1}-1$). Therefore, we propose to employ a single $(n-k)$ -bit counter to realize the run-time update of the control bit values of A_j , cl_j and cr_j . In summary, the total number of latency of the bit-reversal circuit denoted as $T(n, k)$ can be calculated as:

$$T(n, k) = 2^{n-k} \quad (8)$$

Furthermore, the throughput of the design $Th(n, k)$ and the total number of control bits $Ctrl(n, k)$ are:

$$Th(n, k) = 2^k, Ctrl(n, k) = n - k \quad (9)$$

Although regular control bits such as buffer read or write enable bits are not considered in the calculation for $Ctrl(n, k)$, the result still covers the major portion of the control logics and indicates the optimum of the control mechanism used in our designs. With regarding to resource consumption, the number of 2-to-1 multiplexers $Mu(n, k)$ and the number of data buffers $Bu(n, k)$ are:

$$Mu(n, k) = 2k \times 2^k, Bu(n, k) = 2^k \quad (10)$$

4.3 Resource consumption summary

Table 3 shows comparison of several bit-reversal designs. N and p represents the input size and the data parallelism, respectively. Highly memory efficient designs for bit reversal are proposed in [4] and [14], which consume $(\sqrt{N}-1)^2$ and $(5N/8-3)$ memory words, respectively. However, these two designs only support serial input data. To support parallel samples, a parallel-8 bit reversal design is developed in [23] using 16 memory banks having totally $8N$ memory words; such design is able to offer high throughput while not resource efficient. To realize parallel bit reversal on continuous data flows, a design using $2p$ single-port data buffer of total size N and $(2p^2-p)$ 2-to-1 multiplexers is proposed in [11]. Only parallel-2/4/8 designs are presented in this paper. To further reduce the total memory space, a parallel- 2^k design is developed in [12] by expanding the technique proposed in [14]. Their design achieves the lower bound of the number of memory words at the expense of more than $2p \log_2 N$ memory banks (data buffers) and $2p \log_2 N$ 2-to-1 multiplexers, which is resource and power inefficient for large size of N . Besides, the lower bound is hold only when $p < \sqrt{N}$. A more generic approach for parallelizing a class of bit-index permutations including bitreversal is presented in [20]. Permutation matrix manipulation techniques are employed to achieve the lower bound of the number of memory words for realizing arbitrary parallel- 2^k ($k < n$) bit reversal. Al-

Table 4: Resource consumption comparison

p	N	Ours			SPIRAL [20]		
		Slice LUT	Slice Register	BRAM Tiles	Slice LUT	Slice Register	BRAM Tiles
$p=2$	1024	120	16	1	137	42	2
	2048	120	17	2	145	46	4
	4096	134	18	4	152	50	8
	8192	141	19	8	174	54	16
	16384	146	8	16	180	84	32
$p=4$	1024	278	144	2	424	313	2
	2048	279	145	2	430	319	4
	4096	284	146	4	432	325	8
	8192	293	147	8	444	331	16
	16384	302	148	16	462	363	32
$p=8$	1024	917	272	4	1061	1099	4
	2048	918	274	4	1068	1107	4
	4096	919	274	4	1075	1115	8
	8192	921	276	8	1077	1124	16
	16384	939	282	16	1083	1151	32
$p=16$	1024	2616	562	0	2605	3148	8
	2048	2069	535	8	2611	3154	8
	4096	2070	536	8	2614	3161	8
	8192	2089	554	8	2620	3169	16
	16384	2091	552	16	2626	3196	32

though many more recent optimizations have been proposed for reducing memory consumption to less than N , the approach in [20] still remains the most practical one for hardware implementation. However, to support continuous data flows, the memory space requirement in their designs has to be doubled with using dual-port memory banks. Our approach translates the parallel bit reversal problem into a switch network routing problem, and achieves the lower bound of memory words using p single-port memory banks for arbitrary 2^k ($k < n$), simultaneously supports processing continuous data flows. The proposed parallel designs are realized using optimal control mechanism achieving the lower bound of the number of control registers, which is a $(n-k)$ -bit counter. Note that all previous parallel- 2^k designs needs at least a $(n-k)$ -bit register for memory addressing.

5. EXPERIMENTAL RESULTS

To illustrate the benefits of using our proposed parallel designs for bit reversal, we performed detailed experiments on Virtex-7 FPGA (XC7VX690T) by Xilinx Tool Set Vivado 15.2. We choose 32-bit fixed point data vectors as input. The Verilog implementations of the proposed designs and the baselines are available through [1] and [2], respectively. Table 4 shows a resource consumption comparison between our proposed design and the baseline implemented using the technique developed by the SPIRAL project in [20], where a

Figure 8: Energy efficiency comparison of radix-2 FFT

significant amount of efforts have been made to minimize the number of multiplexers and memory consumption. For various p ($p=2,4,8,16$), all our designs beat the baseline with regarding to resource consumption. For $p=2$ and various N , the number of Slice LUTs, Slice registers and BRAM tiles have been reduced by 12%-19%, 62%-90%, and 50%, respectively. For $p=4$ and various N , the number of Slice LUTs, Slice registers and BRAM tiles have been reduced by 34%-35%, 54%-59% and 0%-50%, respectively. When $N=1024$, $p=4$, the data buffer sizes in both our design and the baseline are very small such that only two BRAM tiles are consumed. For $p=8$, the number of Slice LUTs, Slice registers and BRAM tiles have been reduced by 13%-14%, 74%-75% and 0%-50%, respectively for various N . For $p=16$ and $N=2048, 4096, 8195, 16384$, the number of Slice LUTs, Slice registers and BRAM tiles have been reduced by 20%-21%, 82%-83% and 0%-50%, respectively. For $p=16$ and $N=1024$, no BRAM tile is used in our design since distributed RAM composed of Slice LUTs has been employed for implementing small size data buffers.

We further evaluate the energy efficiency (throughput by power) of radix-2 FFT for $N=16, 256, 1024, 2048$ and 4096 while varying p . The operating frequency is fixed at 250 MHz for the sake of power evaluation. The input test vectors were randomly generated with an average toggle rate of 25% (pessimistic estimation). We used the VCD file (value change dump file) as input to Vivado Power Analyzer to provide accurate power dissipation estimation [3]. All the designs were pipelined to achieve this clock rate. The experimental results are presented to demonstrate the benefit of the parallel bit reversal techniques incorporated in our FFT designs from a power point of view. Fig. 8 shows that for small size of N , highly data parallel designs can achieve extremely high energy efficiency. It also shows that as p , and problem size are varied, our design achieves 21% to 49% improvement in energy efficiency compared to [20].

6. CONCLUSION

This paper represents optimal circuit designs for parallel bit reversal. The designs achieve optimal control mechanism and near optimal resource efficiency for calculating continuous-flow bit reversal. The proposed designs are generic and supports arbitrary problem size and data parallelism which are powers of two. Finally, the proposed designs demonstrate significant performance improvement in resource efficiency and energy efficiency compared with the state-of-

the-art, thus are very suitable for reordering the samples of multi-path parallel FFT architectures.

7. REFERENCES

- [1] Hardware Generator for designs in this paper. <http://www-scf.usc.edu/~renchen/>.
- [2] SPIRAL Hardware Generator. <http://www.spiral.net/hardware.html>.
- [3] XST user guide for Virtex-6, Spartan-6, and 7 series devices. <http://www.xilinx.com/support/documentation>.
- [4] M. Ayinala, M. Brown, and K. K. Parhi. Pipelined parallel fft architectures via folding transformation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(6):1068–1081, June 2012.
- [5] V. E. Beneš. Optimal rearrangeable multistage connecting networks. *Bell System Technical Journal*, 43(4):1641–1656, 1964.
- [6] G. Bi and E. Jones. A pipelined FFT processor for word-sequential data. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(12):1982–1985, 1989.
- [7] R. Chen, H. Le, and V. K. Prasanna. Energy efficient architecture for stride permutation on streaming data. In *Proc. of IEEE International Conference on ReConFig*, Dec 2013.
- [8] R. Chen and V. Prasanna. Automatic generation of high throughput energy efficient streaming architectures for arbitrary fixed permutations. In *Proc. of IEEE FPL*, pages 1–8, Sept 2015.
- [9] R. Chen and V. K. Prasanna. Optimizing interconnection complexity for realizing fixed permutation in data and signal processing algorithms. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–9, Aug 2016.
- [10] R. Chen, S. Siriyal, and V. Prasanna. Energy and memory efficient mapping of bitonic sorting on fpga. In *Proceedings of ACM/SIGDA International Symposium on FPGa*, pages 240–249.
- [11] S. G. Chen, S. J. Huang, M. Garrido, and S. J. Jou. Continuous-flow parallel bit-reversal circuit for mdf and mdc fft architectures. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(10):2869–2877, Oct 2014.
- [12] C. Cheng and F. Yu. An optimum architecture for continuous-flow parallel bit reversal. *IEEE Signal Processing Letters*, 22(12):2334–2338, Dec 2015.
- [13] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.
- [14] M. Garrido, J. Grajal, and O. Gustafsson. Optimum circuits for bit reversal. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 58(10):657–661, Oct 2011.
- [15] M. Garrido, J. Grajal, M. Sánchez, and O. Gustafsson. Pipelined radix-feedforward fft architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(1):23–32, 2013.
- [16] B. Gold, T. G. Stockham, A. V. Oppenheim, and C. M. Rader. Digital processing of signals. 1969.
- [17] S. He and M. Torkelson. A new approach to pipeline FFT processor. In *Proceedings of IPSP '96*, pages 766–770.
- [18] T. Jarvinen, P. Salmela, H. Sorokin, and J. Takala. Stride permutation networks for array processors. In *Application-Specific Systems, Architectures and Processors, 2004. Proceedings. 15th IEEE International Conference on*, pages 376–386, 2004.
- [19] T. K. Moon and W. C. Stirling. *Mathematical methods and algorithms for signal processing*, volume 1. Prentice Hall New York, 2000.
- [20] M. Püschel, P. A. Milder, and J. C. Hoe. Permuting streaming data using rams. *Journal of the ACM*, 56(2):10:1–10:34, 2009.
- [21] J. M. Rius and R. D. Porrata-Doria. New fft bit-reversal algorithm. *IEEE Transactions on Signal Processing*, 43(4):991–994, Apr 1995.
- [22] D. Sundararajan, M. O. Ahmad, and M. Swamy. A fast fft bit-reversal algorithm. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 41(10):701–703, 1994.
- [23] S. Yoshizawa, A. Orikasa, and Y. Miyayaga. An area and power efficient pipeline fft processor for 8x8 mimo-ofdm systems. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2705–2708, May 2011.