# Efficient VLSI Implementation of Radix-8 FFT Algorithm

Lihong Jia, Yonghong Gao and Hannu Tenhunen
Electronic System Design Laboratory, Royal Institute of Technology,
Isafjordsgatan 22, Box 229, 16440 Kista, Stockholm, Sweden
Email: jialh@ele.kth.se

**ABSTRACT:** *High-radix Cooley-Turkey FFT algorithms have obvious advantages: less multiplications and reduced memory accesses so power consumption can be reduced. However, the disadvantages are that traditional direct mapping implementation of high-radix butterfly element will required more complex multipliers and thus large silicon area will be consumed. In this paper, we proposed an efficient approach to realize the high radix butterfly process element. This approach employed pipelining technics to cascade the paralleled multipliers and thus fewer complex multipliers are utilized to realize the radix-r butterfly element. This approach can achieve a good trade-off between speed and area in the design of high radix butterfly element.*

*Key word-- high radix, pipeline, direct mapping, butterfly element*

## I. INTRODUCTION

Fast Fourier transform (FFT) operation is one of the most important fundamental operations in the digital signal processing systems. The key problem of today's FFT hardware designer is to reach the given throughput while consuming a minimal amount of power and a reasonable silicon area. [1]

The number of multiplications and memory accesses dominate the power consumption of FFT computation. Its power consumption can be reduced significantly by using high-radix Cooley-Turkey algorithms because the high radix algorithm can reduce the number of multiplications and also can reduce memory access times. [2] [3] [5][7] However, these advantages still did not be effectively utilized because the high-radix butterfly element will consume very large silicon area. The traditional hardware implementation of the butterfly element is direct mapping from the data flow to hardware. This approach is very simple but it is only suitable for the radix-2 or radix-4 butterfly element. For high radix butterfly element, the implementation of direct mapping will consume too large area. Some literatures [3] have addressed the hardware implementation of high radix butterfly element where the bit-serial arithmetic are used to reduced the silicon area. However, the silicon area is still large and the complexity of the high radix butterfly element is still remained.

In this paper we addressed an efficient approach to realize a high-radix butterfly element in a small area but maintaining high throughput. In this approach, the pipelining technic has been used to cascade the high radix butterfly element and fewer multipliers are used instead of r multipliers when radix-r algorithm is used. As a consequence, the circuit complexity of the butterfly element is reduced significantly and at the same time the silicon area is decreased dramatically. As a design example to support the proposed approach, a radix-8 butterfly element is realized in 0.6um CMOS process.

The proposed pipelining implementation of the high radix butterfly element have such advantages: (1) consuming small area (2) achieving high throughput (3) using simple control.

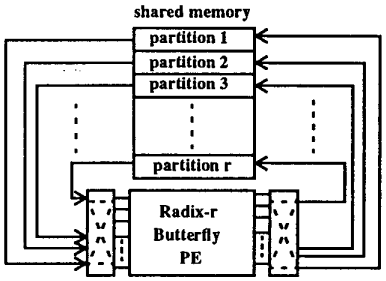
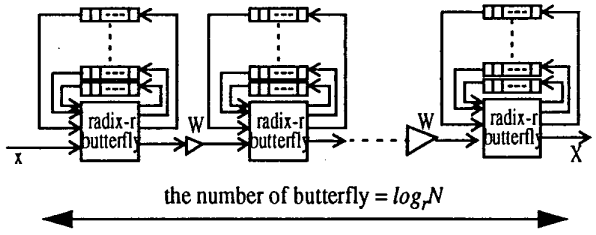
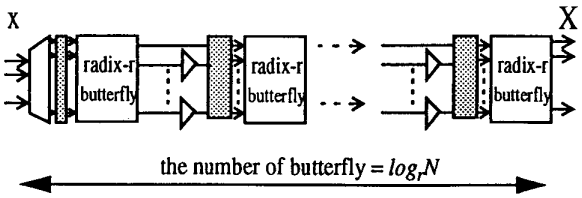



Fig.1 radix-r single-process shared memory architecture




(a) single delay path pipelined architecture




(b)multiple delay path pipelined architecture

**Fig.2 radix-r pipelined architecture**

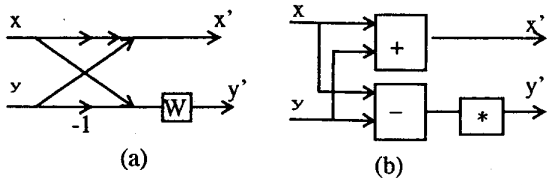



Fig. 3 (a) Radix-2 butterfly structure

(b) the schematic of a radix-2 butterfly element

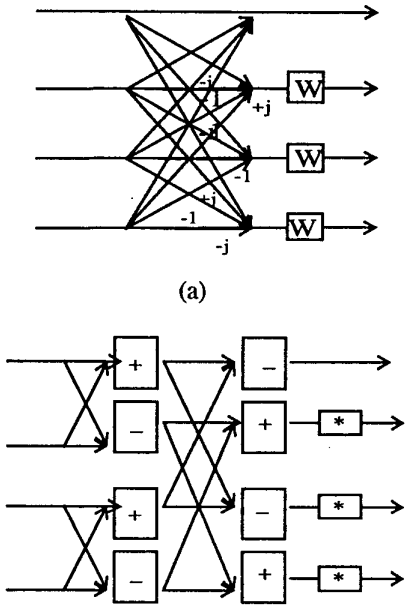



Fig. 4 (a) Radix-4 butterfly structure

(b) the schematic of a radix-4 butterfly element

## II. Radix-R Butterfly Element

Many different architecture for the hardware realization of FFT processors have been developed such as shared memory architecture (shown in fig.1), pipelined architecture (shown in fig.2), array architecture and column architecture. Investigated above architectures, we have such conclusion: (1)The basic element of these architectures is the butterfly element. (2) The basic idea behind these architectures is how to arrange and manage these butterfly elements to effectively compute the N-point FFT operations. As a consequence, the butterfly element is the basic building block in hardware FFT design and the efficient design of the butterfly element is very important to achieve high performance.

$$C_{8k+0}=\sum_{n=0}^{N/8-1}[[(x_n+x_{n+N/2})+(x_{n+N/4}+x_{n+3N/4})]+[(x_{n+N/8}+x_{n+5N/8})+(x_{n+3N/8}+x_{n+7N/8})]]\cdot W^{8nk}$$

$$C_{8k+1}=\sum_{n=0}^{N/8-1}\left[[(x_n-x_{n+N/2})-j(x_{n+N/4}-x_{n+3N/4})]+W^{N/8}\cdot[(x_{n+N/8}-x_{n+5N/8})-j(x_{n+3N/8}-x_{n+7N/8})]\right]\cdot W^{8nk}W^{n}$$

$$C_{8k+2}=\sum_{n=0}^{N/8-1}[[(x_n+x_{n+N/2})-(x_{n+N/4}+x_{n+3N/4})]-j[(x_{n+N/8}-x_{n+5N/8})+j(x_{n+3N/8}-x_{n+7N/8})]]\cdot W^{8nk}W^{3n}$$

$$C_{8k+3}=\sum_{n=0}^{N/8-1}\left[[(x_n-x_{n+N/2})+j(x_{n+N/4}-x_{n+3N/4})]+W^{3N/8}\cdot[(x_{n+N/8}-x_{n+5N/8})+j(x_{n+3N/8}-x_{n+7N/8})]\right]\cdot W^{8nk}W^{3n}$$

$$C_{8k+4}=\sum_{n=0}^{N/8-1}[(x_n+x_{n+N/2})+(x_{n+N/4}+x_{n+3N/4})]-[(x_{n+N/8}-x_{n+5N/8})+(x_{n+3N/8}+x_{n+7N/8})]\cdot W^{8nk}W^{4n}$$

$$C_{8k+5}=\sum_{n=0}^{N/8-1}\left[[(x_n-x_{n+N/2})-j(x_{n+N/4}-x_{n+3N/4})]-W^{N/8}\cdot[(x_{n+N/8}-x_{n+5N/8})-j(x_{n+3N/8}-x_{n+7N/8})]\right]\cdot W^{8nk}W^{5n}$$

$$C_{8k+6}=\sum_{n=0}^{N/8-1}[[(x_n+x_{n+N/2})-(x_{n+N/4}-x_{n+3N/4})]+j[(x_{n+N/8}+x_{n+5N/8})-(x_{n+3N/8}+x_{n+7N/8})]]\cdot W^{8nk}W^{6n}$$

$$C_{8k+7}=\sum_{n=0}^{N/8-1}\left[[(x_n-x_{n+N/2})+j(x_{n+N/4}-x_{n+3N/4})]-W^{3N/8}\cdot[(x_{n+N/8}-x_{n+5N/8})+j(x_{n+3N/8}-x_{n+7N/8})]\right]\cdot W^{8nk}W^{7n}$$

**Fig.5 Mathematic expression of radix-8 butterfly element**

The common approach to implement a butterfly element is direct mapping the structure to the hardware. Fig.3(a) shows the radix-2 butterfly element structure and the schematic of direct mapping of the radix-2 butterfly element. The hardware cost is very cheap: only two complex adder (in two's complement expression, the addition and subtraction have the same hardware cost) and one complex multiplier are needed. We can use the same method to implement the radix-4 butterfly (its structure and schematic are shown in fig.4) where 8 complex adders and 3 complex multipliers are needed. It is still a practical hardware realization although its hardware cost is three times more than radix-2 implementation. When the radice

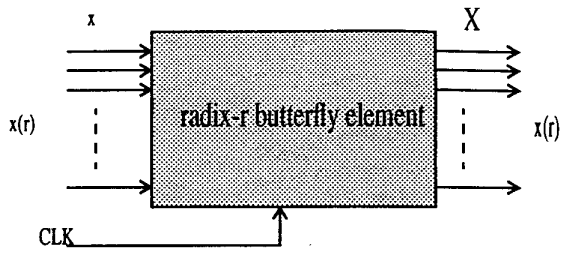
increased, the hardware cost will increase dramatically. Fig.5 shows the mathematic expression for radix-8 from which we can see that the radix-8 butterfly element which required 3 multipliers with -j, 2 multipliers with $(\sqrt{2})/2 \cdot (1 \pm j)$ and 7 non-trivial complex multipliers. In literature [3], a radix-8 butterfly element was designed consuming the silicon area of 21 $mm^2$ in 0.8um CMOS technology although the bit-serial arithmetic was used to reduce the large area.

If radix-r (r>8) algorithm is considered, the situation will be more complicated. Therefore, the bottleneck of high-radix butterfly implementation is how to reduce the hardware area. If the radix r is the power of two, we can use pipelining technics to cascade the butterfly element and thus the adders and multipliers required to computing FFT operations are significantly reduced. Generally pipelined architecture (sometime we also call it cascaded or folding) can achieve the best trade-off between the high throughput and small area. [8] Although the pipelined butterfly element has lower throughput than direct mapping architecture, this loss can be compensated for by using further pipelining between the cascaded butterfly element and inside the butterfly element. Fig.2 shows a single delay path feedback pipelined architecture and multiple delay path pipelined architecture with twiddle factors. In our approach, we use radix-2 pipelined architecture to realize the radix-r butterfly element where r is equal or higher than eight. When the high radix butterfly can be realized in the simple hardware, that means the high radix algorithm can be used to achieve the low power or high speed.

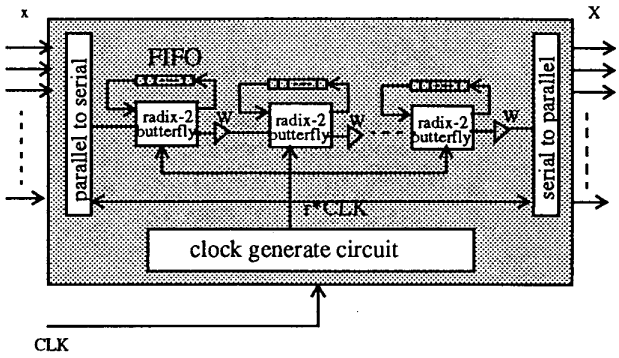
## III. ARCHITECTURE

Fig.6(a) shows the interface of radix-r (r>=8) butterfly element, where the input is the r complex signals from x(0) to x(r-1, the output is r complex signals from X(0) to X(r-1). CLK is the input clock signal. When the CLK comes, the input signals are clocked into the radix-r butterfly element and when next clock comes, the valid output signals are clocked out and the next input signals are clocked in.

Fig.6 (b) shows the block diagram of radix-r butterfly hardware implementation in details which includes four parts. The first part is the parallel to serial and serial to parallel interface which hide the pipelined butterfly element from the rest of the circuits. In the parallel to serial part, the input signals are sampled by an internal high frequency clock which is r times faster than the external input clock. When the computation finished, the internal high frequency serial output signals are converted to parallel low frequency signals. The second part is the clock generator which generates an internal clock which is r times faster than the input clock. The third one is the basic radix-2 butterfly element shown in fig.3, where two adders and one complex multipliers are needed. The fourth one is the FIFO (first in first out buffers) which is used to store the intermediate data. When radix-r butterfly element is implemented, the sizes of FIFOs are r/2, r/4, r/8,.... until 1. For an example, in the radix-8 implementation, we need 3 FIFOs with size of 4,2 and 1, respectively.




(a) radix-r butterfly element interface




(b) radix-r butterfly pipelined implementation

Fig.6 SDF pipelined radix-r butterfly element

## III DESIGN EXAMPLE

Fig.7 (a) shows the signal flow of a radix-8 butterfly element and fig.7(b) shows the SDF pipelined architecture. In this design, the multiplication with j or -j can be done without hardware cost by interchange the real part and image part. As fig.8 has shown, the multiplication with constant twiddle factors $(\sqrt{2})/2 \cdot (1 \pm j)$ can be replaced by 12 additions using shift-adders [4]. Thus complexity of this radix-8 butterfly

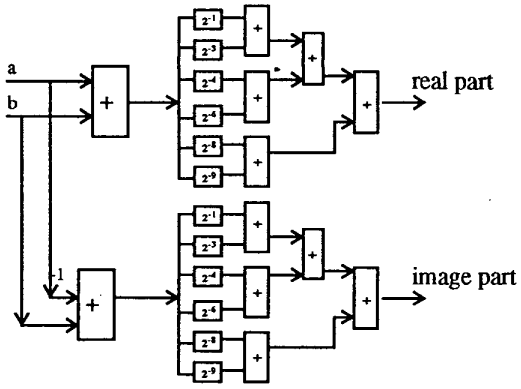
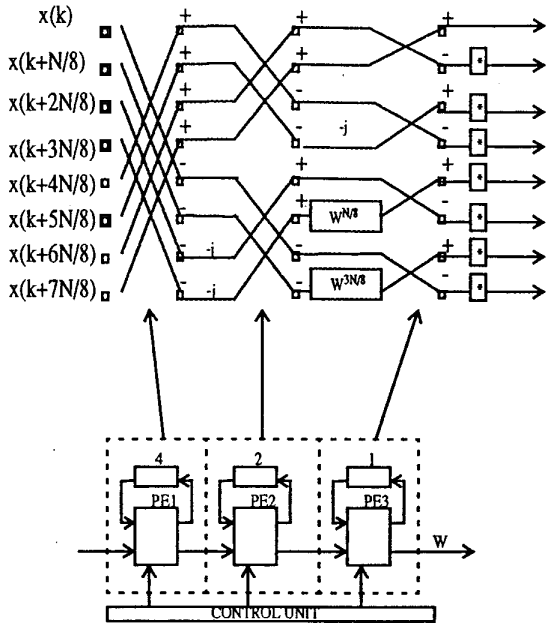



Fig.8 the multiplication with $(\sqrt{2})/2 \cdot (1 + j)$

(a) signal flow graph for radix-8 butterfly




(b) radix-2 SDF pipelined architecture for radix-8 butterfly

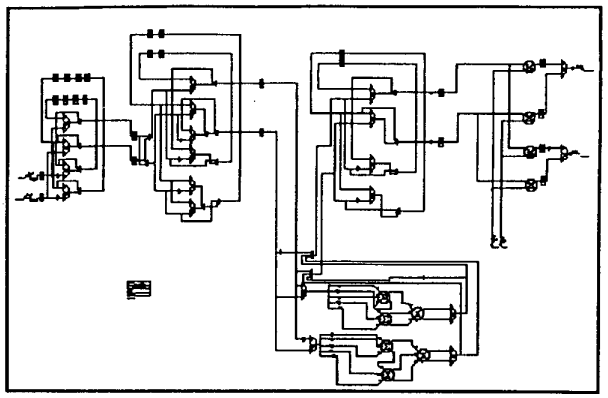
Fig.7 radix-8 butterfly using SDF pipelined architecture

element is dramatically reduced and it only has one complex multiplier and thus its area is very small.

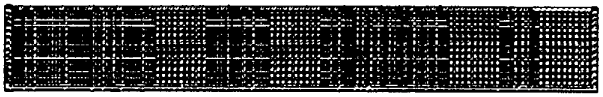
We design such a pipelined radix-8 butterfly element in 0.6 um CMOS technology. Fig.9 shows the datapath description for pipelined radix-8 butterfly element (only including the butterfly elements and FIFOs, the clock generator and parallel to serial and serial to parallel circuits are not included) and its corresponding layout compiled in COMPASS tools.[6] In order to achieve high speed, pipelining is added between stages to speed up the circuits. Pipelining is also used inside the stages. The simulated data rate of pipelined butterfly element is 80MHz. And so the time to compute a radix-8 butterfly is about 0.1 us. It occupied the area of 6.4x0.89 = 0.36 $mm^2$. It is a very small area compared to the direct mapping in [3] and is a very area efficient design.

## V. COCLUSION

In this paper we addressed an efficient approach to implement the high-radix butterfly element. In this approach, we use pipelining technics to cascaded the butterfly element and thus the silicon area for high radix butterfly is significantly reduced. This approach takes the both advantages of high throughput and small area. Therefore the high-radix FFT processor can be efficiently realized using the proposed high-radix butterfly element. A design example has demonstrated its feasibility.



(a) radix-8 butterfly element datapath description



6.4mm x 0.89mm

(b) radix-8 butterfly element layout

Fig.9 radix-8 butterfly element implementation

## VI. REFERENCES


[1] S.He, "Concurrent VLSI Architectures for DFT computing and Algorithms for Multi-output Logic Decomposition", Diss. No. 133, Lund university, Sweden, 1995.

[2] Vacher, A.; Guyot, A., "Radix-8 butterflies for folded FFT", Twenty-Seventh Southeastern Symposium on System Theory, 1995. Page(s):470 -473.

[3] T.Widhe, J.Melander and L. Wanhammar, "Design of Efficient Radix-8 Butterfly PEs for VLSI", IEEE Proc. of International Symp. on Circuits and Systems, on Volume: 3, Page(s): 2084-2087, 1997, HongKong.

[4] Lihong Jia, Yonghong Gao, Jouni Isoaho and Hannu Tenhunen, "A New VLSI-Oriented FFT Algorithm and Implementation", accepted by 11th Annual IEEE International ASIC Conference, September 1998, Rochester, New York, USA.

[5] Bernard, E.; Krammer, J.G., Sauer, M.; Schweizer, R., "A pipeline architecture for modified higher radix FFT", IEEE International Conference on Acoustics, Speech, and Signal 1992. ICASSP-92. Page(s): 617-620 vol.5

[6] "Manual of COMPASS 0.6 μm 3.3 V CMOS Libraries", COMPASS On-line Documents.

[7] Swartzlander, Young, and Joseph, "A radix-4 delay commutor for fast Fourier transform processor implementation", *IEEE J. Solid-State Circuits*. vol. SC-19, Oct. 1984.

[8] E.H. Wold and A.M.Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations", *IEEE Trans. on Computers*, vol.C-33, No.5, pp.414-426, May, 1984.