Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Scalable FFT Architecture vs. Multiple Pipeline FFT Architectures –Hardware Implementation and Cost

Adnan Suleiman, Adel Hussein, Khaldoun Bataineh, David Akopian
Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, Texas USA

Abstract—— **This paper presents a family of architectures for FFT implementation based on the decomposition of the perfect shuffle permutation, which can be designed with variable number of processing elements. This provides designers with a trade-off choice of speed vs. complexity (cost and area.). Hardware comparison to other existing pipeline architecture presented based on implementation of 1024-point FFT with 4 processing elements using 45nm process technology. The proposed architecture is most suitable for handheld and portable multimedia applications**

Keywords—**Scalable, Architecture, FFT, 45nm technology, FIFO** ***(key words)***

I. INTRODUCTION

The Fast Fourier Transform (FFT) is a conventional method for an accelerated computation of the Discrete Fourier Transform (DFT) [1]. FFT has been used in many applications such as spectrum estimation, fast convolution and correlation, signal modulation and many wireless multimedia applications. Even though FFT algorithmically improves computational efficiency, additional hardware-based accelerators are used to further accelerate the processing through parallel processing techniques. Generally, there are two industry standard approaches for implementing FFT in hardware, pipelined and memory-based. A variety of architectures have been proposed to increase the speed, reduce the power consumption and area.

In [2] two N-word memories located at the input and output of FFT processor are used for continuous computation flow. Each partitioned into four banks sharing same FFT computation resources. A multiplexed butterfly used for two radix-2 or one radix-4 FFT computation.

Modular pipeline architecture [3] uses two $\sqrt{N}$ conventional pipelines joint with central memory unit. This architecture reduces the number of delay elements and distributed ROM requirement.

The multidimensional index mapping architecture [4] reduces requirements for memory delay elements based on divide and conquer of twiddle factors multiplications.

A continuous flow mixed FFT (CFMM-FFT) presented in [5,6] performs real-to-complex FFT and complex-to-real IFFT utilizing the same resources, thus minimizing memory requirements. FFT computation carried out over four memory banks and two butterflies.

In [7] a unified approach to counter memory contention is employed via partitioning memory into several banks. The approach increased memory bandwidth that led to pipeline the computations inside butterflies. Modified pipeline architecture is presented in [8] and a radix-2^2 is mapped to radix-4 algorithm as in [9] taking advantage of radix-4 multiplications operations but still remains a radix-2. Another pipelined architecture aimed at minimizing the number of complex computations presented in [10]. One last approach investigated in this paper is a combined feedback and feedforward commutator schemes in [11] thereby achieving 100% utilization.

The method proposed in this paper is based on decomposition of perfect shuffle that results in a constant geometry structures. The available number of processing elements compute each stage data. PEs compute several butterfly operations in parallel and the process is repeated until all butterflies of a stage are computed. Then PEs start computation of the next stage and so on. Data flow between processors is automated guaranteeing correct uninterrupted data flow. This approach allows for a choice of specific number of processing elements thus, the scalable architecture. This work is a continuation of the effort started in [12,13].

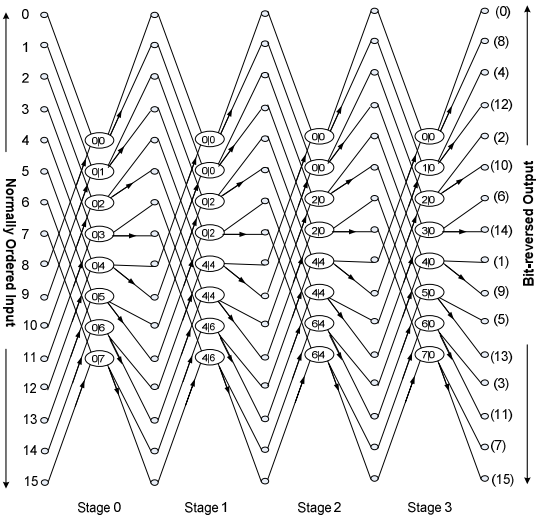

Figure1. Constant Geometry Not-in-place16-point FFT.

II. FFT CONSTANT GEOMETRY

In [1] Radix-2 FFT of N points –where N is integer power of 2- requires N log2N complex operations compared to N^2 of direct DFT computation. Also, FFT algorithms are known in two types, decimation in time (DIT) and decimation in frequency (DIF). The complex multiplication occurs after the two-point DFT for DIT while complex multiplication takes place before the two-point DFT for DIF type algorithm. Further more, the notations "in-place" and "not-in-place" refer to whether an input point ends up in the same register it came from or not. FFT algorithm consists of $\log_2 N$ stages. Each stage consists of N/2 radix-2 butterfly operations. Figure 1 shows a radix-2 16-point for N=16 and W=2 constant geometry algorithm with ordered input and bit-reversed output [14]. Since output of the butterflies did not go back from where it came, this refers to not-in-place algorithm.

III. DECOMPOSITIONS OF PERFECT SHUFFLE (UNSHUFFLE) REORDERING

In [15] data permutations are considered as operators acting on the indices of data where code word for binary representation of an index is specified by the fields:

$$[x,y,z]=[x_u,\ldots,x_1\ y_v,\ldots,y_1\ z_w,\ldots,z_1] \quad (1)$$

Where: $[x_i,y_i,z_i]\in\{0,1\}$ and $x.2^{w+v}+y.2^{v}+z$ is the actual index value.

The representation describes data flow in parallel structure and translated [16] as [cycle, PE, path] where cycle, references the cycle at which data is available for computation, PE is a processing element –otherwise butterfly- and path, the particular input channel of a processing element. If R is a reordering operator acting on an index then applying operator R to this index yields:

$$R[a_{l-1},\ldots,a_0]=[b_{l-1},\ldots,b_0] \quad (2)$$

Applying several iteration of the data permutation sequentially is possible yielding the following:

$$R_2(R_1[a_{l-1},\ldots,a_0])=R_2([b_{l-1},\ldots,b_0])=[c_{l-1},\ldots,c_0] \quad (3)$$

This convention permits the identification of intermediate location during sequential data permutation as can be illustrated in Figure2. Furthermore, let us define the operator acting on indices of data locations and performing a perfect shuffle, $P^T_{N,2}$ – un-shuffle, $P_{N,2}$ - permutation of order k as:

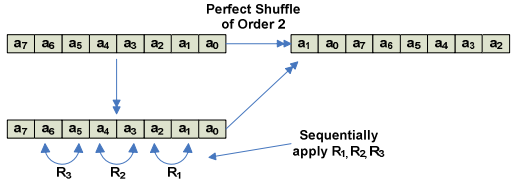

Figure 2. Perfect Shuffle Decomposition.

$$\sigma_{(k)}[x,y,z]=[[z_k\ldots z_1\ x_u\ldots z_{k+1}],\ [x_k\ldots x_1\ y_v\ldots y_{k+1}],\ [y_k\ldots y_1\ z_w\ldots z_{k+1}]] \quad (4)$$

And

$$\sum_{(k)}[x,y,z]=[[x_{u-k}\ldots x_1\ y_v\ldots y_{v-k+1}],\ [y_{v-k}\ldots y_1\ z_w\ldots z_{w-k+1}],\ [z_{w-k}\ldots z_1\ x_u\ldots x_{u-k+1}]] \quad (5)$$

Additionally, these operators could be defined acting on two indices as:

$$\sigma^{x,y}_{(k)}=[[y_k\ldots y_1\ x_u\ldots x_{k+1}],\ [x_k\ldots x_1\ y_v\ldots y_{k+1}],\ [z_w\ldots z_1]]. \quad (6)$$

Then according to [15] the following fact holds:

$$\sigma_{(w)}[x,y,z]=\sigma^{x,z}_{(w)}\ \sigma^{y,z}_{(w)}[x,y,z].$$
$$\sum_{(w)}[x,y,z]=\sum^{y,z}_{(w)}\sum^{x,z}_{(w)}[x,y,z]. \quad (7)$$

Local ordering $\sum^{x,z}_{(w)}$ is implemented by way of FIFO associated with each processing element (PE) for how many are there exist in design. While $\sum^{y,z}_{(w)}$ can be implemented as the interconnect network between PEs represented in external hard connection. In general, the implementation of external interconnection between PEs and sequential reordering network are done by a network of hard connections. Additionally, as mentioned earlier, the local reordering is implemented using special memory organization comprising sequential perfect shuffle and or un-shuffle networks (SPSN).

A. Proposition 1

(The local) perfect un-shuffle or and shuffle reordering of order w could be decomposed into set of reordering R¬i as follow:

$$[\sigma_{(w)}][x^p\ldots x^0]=R^p\cdot R^{p-1}\cdot\ldots R^1\cdot[x^p\ldots x^0],$$
$$[\sum_{(w)}][x^p\ldots x^0]=R^1\cdot\ldots\cdot R^{p-1}\cdot R^p\cdot[x^p\ldots x^0]. \quad (8)$$

Where:

$$x_i=[x^i_w\ldots x^i_1],\ x^j_i\in\{0,1\}\quad and$$
$$R^i[x^p\ldots x^i\ x^{i-1}\ldots x^0]=[x^p\ldots x^{i-1}\ x^i\ldots x^0].$$

B. Proposition 2

$$R^i=R^i_1\cdot\ldots\cdot R^i_{w-1}\cdot R^i_w \quad (9)$$

$$R^i_t[x^p\ldots x^i\ x^{i-1}\ldots x^0]=$$
$$R^i_t[*\ldots*\,x^i_w\ldots x^i_t\ldots x^i_1 x^{i-1}_w\ldots x^{i-1}_t\ldots x^{i-1}_1*\ldots*]=$$
$$[*\ldots*\,x^i_w\ldots x^{i-1}_t\ldots x^{i-1}_w x^i_1\ldots x^i_t\ldots x^{i-1}_1*\ldots*] \quad (10)$$

IV. ARCHITECTURE

Figure 3 shows block diagram of 2-PE radix-2, W=2, system with a perfect shuffle interconnect network and FIFO's. The scalable FFT comprises of number of computational elements (PEs), including complex adders and multipliers, reordering network, and storage elements.

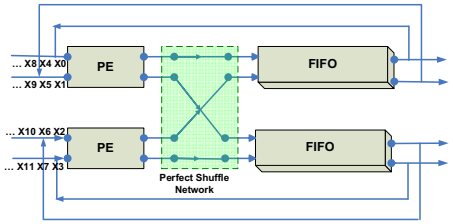

Figure 3. 2PE, W=1 FFT Architecture.

For the purpose of simplicity of our system description, feedback loop connections are shown in here. In this configuration, there are two PEs serve the entire system for the total number of 2^n point in the transform. A PE serves as data feeder for on-the-fly input data. Input data can be from external host or the inter-stages loop data. In addition to this, the PE hosts the complex computational butterfly as shown in Figure 4.

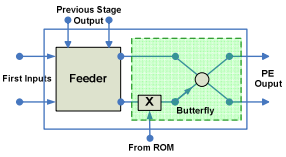

Figure 4. PE Structure

FIFO's comprised of SPSN network, which its shift-exchange units (SEU) are proportional to the size of FFT computational points N. For the scalable architecture, first, the number of PEs is decided then FIFO size becomes totally dependant on the FFT size N.

The number of SEUs in a FIFO increases dramatically with large input N. For example, for an N = 1024 and PE =2 radix-2 FFT, the number of SEUs reaches 8 with last SEU holding the most delay storage of 128.

Butterflies perform complex addition and multiplication operations; thus, for radix-2 FFT, there are four real adders and 6 real multipliers. Our RTL model is based on fixed-point operations in which, real and fractional parts of a data point are split in half. Depending on configuration, the output of a PE is sent to either reordering interconnect network or FIFOs

A third component of major importance is the perfect shuffle (unshuffle) network. As mentioned earlier, this part of the architecture is implemented with direct connections either prior to FIFO inputs forming (perfect shuffle) or after FIFO outputs forming (perfect unshuffle). Both configurations (perfect shuffle/unshuffle) facilitate external data reordering in coordination with FIFO's SEUs forming an uninterruptible data exchange and reordering system.

Note that Butterflies and interconnect network are configured similar to that of Figure 4 and used throughout the rest of this paper.

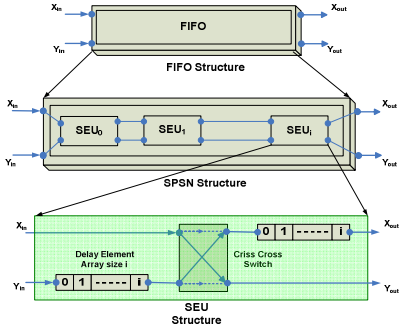

Figure 5. FIFO Structure.

Figure 5 shows top-down decomposition structure of a FIFO and its components. A FIFO is formed by way of SPSN that essentially is composed of number of smaller units called SEUs. The number of SEUs in an SPSN is calculated as

$$\#SEU = \log_2(\frac{N}{PE}) - 1 \qquad (11)$$

Equation (11) says that when the number of PEs increases, the number of delay elements is reduced dramatically. This in tern decreases overall system delay. Table 1 presents SEUs calculation for various N samples.

TABLE I. CALCULATION OF NUMBER OF SEUS IN A DESIGN.

N	2PE	4PE	Num Stages
16	2	1	4
32	3	2	5
64	4	3	6
1024	8	7	10

Generally, SEUs have one common structure; two equally sized arrays of delay elements separated by cress-cross switch – also known as commutator in other structures. However, they differ by the size of delay arrays, which in their part dependant on the SEU sequence order (i) in the SPSN. Therefore, SEUs are sized in the order of 2^0, 2^1, 2^2, 2^3… 2^i.

The operation of SEU is simple; an element entering an SPSN is delayed by the length of the delay array of current SEU from its coupled input at the second SPSN. Data shuffle and reordering occurrence is based on the status of current switch –commutator- and current location of data datum in the delay arrays. Reordering occurs during SEU's switch on position at which, an element moves from the first SPSN to the other and vise versa. The output of an SEU is sequentially fed into subsequent SEUs with longer delay and slower

switching rate according to a ratio of order 2^0, 2^1, 2^2, 2^3… 2^i. The total delay in FIFO is the sum of delays in all SEUs and the total delay elements in the entire structure is the sum of delays in all FIFOs.

For illustration purposes, Figure 6 shows an example for 16-point FFT with 2-PE. The structure has two SEUs based on (11) with delays of one and two respectively. The transform will have four stages. In each cycle, the structure can process four data points. This means that we need four cycles per stage

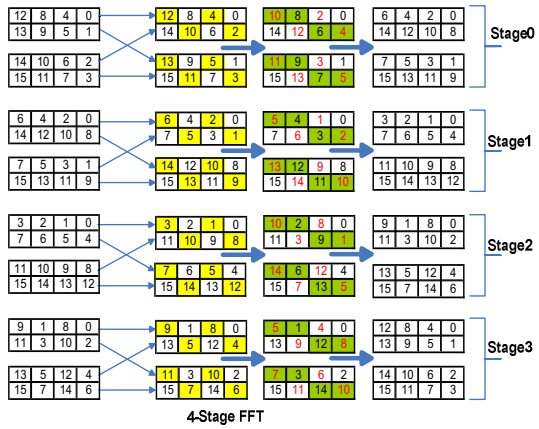

Figure 6. PE=2, W=1, N=16, All Stages FFT .

to process all 16 points. For simplicity, location indexes are used instead of actual data. This allows for monitoring reordering and shuffling of any data point. First data input through feeder out to interconnect then into SEU0 (yellow) and out of SEU1 (green). The process repeats for four stages as the output of current stage is the input of the next.

Figure 7 shows the first stage of 64-point FFT transfer of 4PE-design. Total cycles to calculate one stage of the 64-point for 2PE and 4PE is 16 and 8 cycles respectively and overall calculation time is 6 x 16 + 15 = 111 cycles for 2PE and 6 x 8 + 7 = 55 cycles for 4PE designs.

As an observation from this example (increasing PE

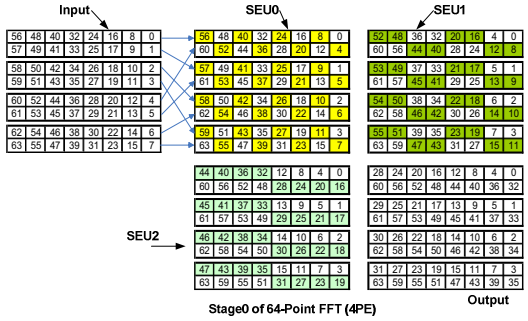

Figure 7. PE=4, W=1, N=64, Stage0 FFT.

number), the complexity of design increases in two areas: first the area of processing elements; and second, doubling the size of perfect shuffle interconnect network –mainly datapath sizing. Additionally, number of clock cycles for 64-point FFT computation is reduced by more than 50% contributing to increase in throughput (will double) of about the same factor as well.

Other performance parameters are in one way or the other impacted by the total number of PEs available in a design as follows: number of coefficients storage is N/PE, overall delay equals to N/PE, and total required cycles is (N/2PE)*$\log_2$N – 1. Each butterfly requires 4 multipliers and 6 adders for complex operation, thus, 4*PE real multipliers and 6*PE real adders are required in a design. The cost of the multipliers is very expensive when it comes to area complexity. Number of FIFOs in a design is equal to number of PEs. Number of SEUs in a FIFO is calculated by (11). Number of switches in a design is equal to PE * SEUs in a FIFO. Additionally, as a comparative edge, tables 2, table 3 and table 4 show summaries of important architecture characteristics comparison with other architectures –discussed in introduction- and the scalable architecture.

TABLE IV. CONVENTIONAL PIPELINE AND SCALABLE ARCHITECTURES CHARACTERISTICS COMPARISON.

Arch.	Processing Time	2048 Points	4096 Points
[5]	N*$\log_2$(N)/2	11264	24576
[7]	N*$\log_2$(N)/2+2	11266	24578
[6]	-	6144	12288
[2]	N*$\lceil \log_4(N) \rceil$/4+2	3074	6146
[3]	N + $N^{1/2}$-2	2091	4160
Scalable	(N/2PE) $\log_2$N -1	2815	6143

TABLE II. SCALABLE ARCHITECTURE AND OTHER ARCHITECTURE HARDWARE COMPARISON.

Arch.	Adders Complex	Multipliers Complex	Coefficients Storage
[8]	3 $\log_4$N	$\log_4$N-1	N
[9]	2 $\log_2$N	$\log_4$N-1	N
[10]	2 $\log_2$N	$\log_8$N-1	N
[11]	8($\log_4$N+1)	3($\log_4$N-1)	N/2
[3]	16 $\log_2 N^{1/2}$	2 $\log_2 N^{1/2}$-1	N+2$N^{1/2}$
[4]	2 $\log_2$N	$\log_8$N-1	3N/8
Scalable	2PE	2PE	N/PE

TABLE III. CONVENTIONAL PIPELINE AND SCALABLE ARCHITECTURES CHARACTERISTICS COMPARISON.

Factor	Pipeline Architecture	Scalable Architecture
Coefficient Storage	N - 2	N / PE
Delay Elements	N – 2	N / PE
Multipliers (real)	4 $\log_2$N	4 PE
Adders (real)	6 $\log_2$N	6 PE
Total time to process first N point	N/2 cycles	N/2PE – 1
Subsequent N points	N/2 cycles	N/2PE

V. IMPLEMENTATION & VERIFICATION

The FFT core was designed using Verilog-HDL and implemented using an automatic synthesizes place and route approach; see [17]. A very high performance, 45nm, technology process was used for the implementation with a high performance standard cell library carefully designed for high-speed applications. The routing was limited to metal layer-7. Table 5 summarizes implementation results. The design met timing at 653.6 MHZ. At this cycle speed, a 1024-point FFT using four PEs completes in ((1024/2*4) - 1 for first stage + (1024/2*4) * 9 for the rest of stags) = 1279 cycles. At a 1530ps cycle time, this translates to 1279* 1.53ns = 1.957µs.

TABLE V. DESIGN SPECIFICATIONS.

Item	Details
FFT Algorithm	Radix-2, Decimation-in-Frequency
N	1024 points
Format	Fixed-point (int.frac): 16.16
Number of PEs	4
Total Number of Cells	89,045
Combinational area %	56.3%
Non Combinational area %	43.7%
Height	569.52 uM
Width	570.08uM
Utilization	69.3%
Total Wire Length	1,937,640.81 uM
Frequency	653.6 MHz
Technology	45 nm Bulk CMOS
Supply Voltage	0.9 V
Dynamic Power	168.6mW
Leakage Power	14.7mW

VI. CONCLUSION

The proposed systematic scalable pipeline architecture presents a new efficient method for decomposition of perfect shuffle permutation and data reordering for FFT algorithm. This structure provides a tool for designers to evaluate design complexity versus throughput. Examples for both 2 and 4 PE radix-2 FFT discussed in detail. A 1024 points radix-2 with four PEs implementation results presented as prove of concept. The structure compares favorably, in terms of speed and area, with other structures including the pipeline one. Proposed architecture can be easily proved extendable to higher FFT radix.

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

[2] J. Wu; K. Liu; B. Shen & H. Min, "A hardware efficient VLSI architecture for FFT processor in OFDM systems," ASIC, 6th Int. Conf. vol. 1, pp. 232-235, 2005.

[3] A. M. Al-Khashab, E. E. Swartzlander, "An architecture for a radix-4 modular pipeline fast Fourier transform," IEEE Aplication- Specific Systems, Architectures, and Processors, Conf., pp. 378-388, June, 2003

[4] Xin Fan, "A VLSI-oriented FFT algorithm and its pipelined design," Int. Conf. on Signal Processing, 2008, pp. 414-417.

.

[5] R. Radhouane, P. Liu, and C. Modlin "Minimizing the memory requirements for continuous flow FFT implementation: continuous flow mixed mode FFT (CFMM-FFT)," in Proc. IEEE Int. Symp. Circuit Syst., 2000, Vol. 1, pp. 116-119.

[6] Amphion, CS2420 2048/4096/8192 Point FFT/IFFT, April 2002

[7] H.F Lo, M.D Shieh, and C.M Wu, "design of an efficient FFt processor for DAB system," in Proc. IEEE Int. Symp. Circuit Syst., 2001, Vol. 4, pp. 654-657.

[8] G. Bi and E. V. Jones, "A pipelined FFT processor for word-sequential data," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, pp. 1982-1985, December 1989.

[9] He, S. and M. Torkelson. "Design and Implementation of a 1024-point Pipeline FFT Processor," IEEE Custom Integrated Circuits Conference, pp. 131–134, May 1998.

[10] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunmen, "A new VLSI-oriented FFT algorithm and implementation," Eleventh Annual IEEE International ASIC Conf., pp. 337-341, Rochester, NY, Sept. 1998

[11] Y. N. Chang, Keshab K. Parhi, "An efficient pipeline FFT architecture," IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing, vol. 50, No. 6, pp. 322-325, June 2003.

[12] D. Akopian, J. Takala, J. Astola and J. Saarinen, "Multistage interconnection network for k/n rate parallel Viterbi decoders," IEEE Trans. on Communications, Sep. 2003

[13] D. Akopian and J. Astola, "Fast architecture oriented algorithms for trigonometric transforms and their mapping to scalable structures," Proceedings of First International Workshop on Transforms and Filterbanks, TICSP Series-1, June 1998, pp. 433-471.

[14] Rabiner, L. R. and Gold, B."Theory and Application of Digital Signal Processing," Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[15] E. L. Zapata and F. Arguello, "A VLSI constant geometry architecture for the fast Hartley and Fourier transforms," IEEE Trans. Parallel Distrib. Syst.,vol.3, pp. 58-70, 1992.

[16] E. L. Zapata and F. Arguello, "Application-specific architecture for fast transforms based on successive doubling method," IEEE Trans. Sig. Processing, vol.41, pp. 1476-1481, 1993.

[17] Suleiman, A.; Saleh, H.; Hussein, A.; Akopian, D.;" A family of scalable FFT architectures and an implementation of 1024-point radix-2 FFT for real-time communications,"IEEE, ICCD2008, pp. 321-327