Dual-Purpose Hardware Accelerator to implement a High-throughput FFT and Sorting Engine

Indu Prathapan
Embedded Processing
Texas Instruments
Bangalore, India
induprathapan@ti.com

Pankaj Gupta
Analog Signal Chain
Texas Instruments
Bangalore, India
pankajgupta@ti.com

***Abstract*—This paper describes a novel architecture for sorting binary numbers in hardware, based on a Radix-2 single delay feedback (R2SDF) architecture that is popularly used to implement pipelined Fast Fourier Transform (FFT) processors. The sorting algorithm used in this implementation is bitonic sorting. The spatial regularity of the bitonic sorting network and its similarity to FFT's signal flow graph (SFG) is exploited to map its comparator stages to the pipelined stages of the R2SDF FFT hardware with negligible area increase as compared to the original FFT engine. The data-path components like radix-2 butterfly units in a R2SDF FFT are replaced with 2-input comparators and some additional control logic for sorting engine. The proposed hardware accelerator can be configured by software to either compute FFT of N data elements or sort the N elements in linear order. For sorting N binary numbers fed into the proposed sorting engine in a serial order, a total of $\Theta(N*\log_2 N)$ clock cycles are required. This is equal to the theoretical upper bound for sorting speed achievable with any comparison based sorting algorithm. The throughput of the serial hardware accelerator is further improved by 4x by increasing the parallelism in both the FFT engine and the Sorting engine. The proposed architecture is implemented in 45nm CMOS technology, meeting 400 MHz clock frequency.**

Keywords—sorting, bitonic, R2SDF, FFT, pipelined architecture

I. INTRODUCTION

The Fast Fourier Transform (FFT) operations that are commonly used in digital signal processing have a regular structure that makes it an ideal candidate for direct implementation in hardware accelerators. The hardware accelerators offload common signal processing operations that are computationally-intensive from the core processor and thus, allowing it to focus on other general-purpose tasks. This approach boosts up the effective computational throughput of the processor and hence, the overall system performance.

Sorting is also one of the key functions performed by computer programs as an internal step for many data processing and computer graphics applications. Many algorithms used in signal processing applications need input data to be in sorted order. There are numerous, well researched, fast and efficient sorting algorithms described in literature[1][2], but practically the main bottleneck for implementing sorting on a processor involves memory access latency, limited bandwidth, and memory contentions that are completely system dependent.

This paper focuses on an implementation for sorting using a hardware accelerator. Radix-2 single-delay-feedback (R2SDF) is a popular pipelined implementation for FFT accelerator design that is optimized with respect to memory structure, control units, and processing elements [4][5]. In the proposed hardware sorting implementation, the well-known bitonic sorting network[3] is adopted and mapped to the R2SDF FFT structure. This allows us to re-use most of the R2SDF hardware logic with the exception of data-path elements like butterfly units and multipliers which get replaced by sorting specific hardware units like compare and exchange units.

We present a detailed description of the design and implementation of a dual-purpose sorting and FFT accelerator that are very relevant to modern day systems that needs both FFT and sorting operations for its algorithms to implemented. We also describe a method to improve the throughput of the system by introducing a higher degree of parallelism into the hardware accelerator engine. Although this paper refers to a dual-purpose FFT and Sorting accelerator, the proposed architecture can also be used to implement a stand-alone sorting accelerator, with performance that matches the theoretical upper bound using any comparison based sorting algorithm, consuming a modest on-chip memory of size N data words. Throughout this paper, we use N to denote the size of a real-data sequence to be sorted and also to denote the size of FFT. Without losing generality, we assume N is a power of two.

The rest of this paper is organized as follows. In Section II, we discuss the bitonic sorting algorithm. We, then, discuss R2SDF FFT architecture in Section III. In Section IV, we propose our dual-purpose hardware (HW) accelerator architecture in detail. Section VI provides HW implementation complexity of the proposed solution. Finally, we present our conclusions in Section VII.

II. BITONIC SORTING ALGORITHM

A bitonic sequence is a sequence of elements $(a_0, a_1, \ldots, a_{N-1})$ that satisfies either of the below two conditions

1. There exits an index i, $0 \leq i \leq$ N-1, such that $(a_0, \ldots, a_i)$ is monotonically increasing and $(a_{i+1}, \ldots, a_{N-1})$ is monotonically decreasing.
2. There is a cyclic shift of indices so that (1) is satisfied.

For example, {1, 4, 6, 8, 3, 2}, {6, 9, 4, 2, 3, 5} and {9, 8, 3, 2, 4, 6} are bitonic sequences.

Given a bitonic sequence of size N, if we divide the sequence into two halves and do a compare-and-exchange operations for each of the elements a_i and $a_{i+N/2}$, we get two bitonic sequences in which all the values in one sequence are smaller than the values of the other. Applying these operations to a bitonic sequence recursively, we get a monotonically sorted sequence. Bitonic sorting algorithm takes advantage of this property of the bitonic sequences to

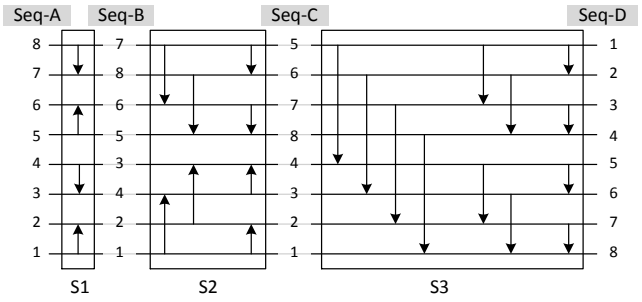


Fig.1 : Bitonic Network for N = 8

generate a parallel sorting network [3].

Fig.1 shows the signal flow graph of a Bitonic network for sorting a data sequence of size 8 with random inputs. The arrows indicate the two elements to compare (located at the head and tail of each arrow) and the direction to swap them such that we end up with the smallest element at the tail.

A bitonic sorting network consists of the following two operations:

1. Rearrangement of an unsorted data sequence (Seq-A) into a bitonic sequence (Seq-C). This is performed in the first $\log_2 N - 1$ stages.
2. Rearrangement of the Bitonic sequence(Seq-C) into a sorted sequence (Seq-D) is performed in the last stage.

Input data (Seq-A) is considered to be bitonic sequence of length 2. With parallel compare-and-exchange operations in opposite directions for adjacent sequences in first stage (S1), pairs of adjacent numbers are converted into bitonic sequences of length 4 (Seq-B). Subsequent stages continue with the same operations until a bitonic sequence of length N (Seq-C) is generated. The last stage (S3) then converts the bitonic sequence Seq-C into a sorted sequence Seq-D.

A parallel implementation of bitonic sorting network sorts N elements in $\Theta(\log_2 N)$ clock cycles using $\Theta(N*\log_2 N)$ comparators, assuming that the outputs of all compare-swap units are pipelined. However, the availability for N elements at the same time is not practically possible in a system, especially if N is large. Hence, we will target sorting N elements in $\Theta(N*\log_2 N)$ clock cycles, that is equal to the worst case comparisons required to sort N elements in any comparison sort[2].

III. Radix-2 Single Delay Feedback (R2SDF) FFT Architecture

This is a well-known structure for the implementation of pipelined FFT processors [4][5]. It consists of $\log_2 N$ pipelined butterfly stages where N is the number of FFT points. Delay elements using memory or shift registers are present in the feedback path of the butterfly units in all stages. Signal flow graph (SFG) of an 8-point FFT and its mapping to an 8-point R2SDF FFT structure is illustrated in Fig. 2. Every stage of the SFG maps to one of the pipeline stages in R2SDF. Delay-lines (or feedback path) of length 2m are required for each of the m stages, where m ranges from 0 to $\log_2 N - 1$. The number of delay elements in each stage is equal to vertical "distance" between the

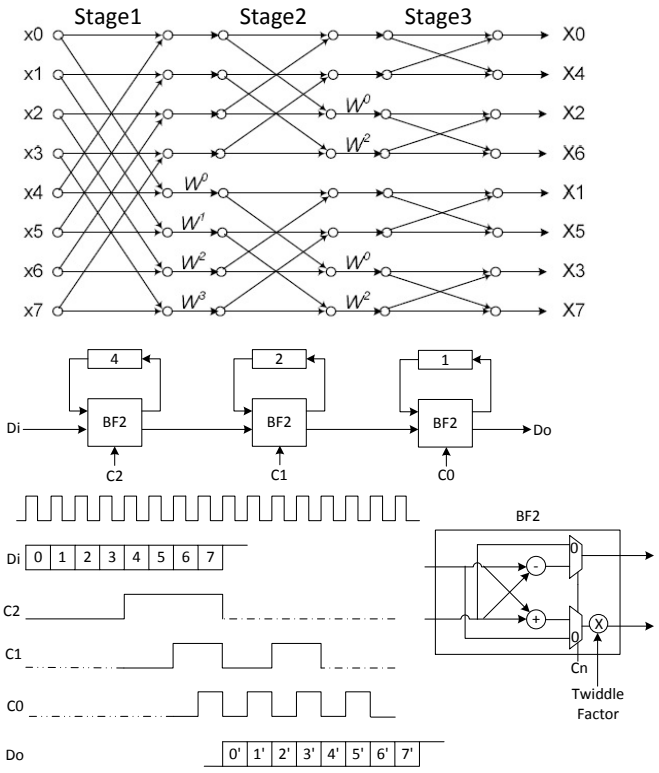


Fig. 2 : R2SDF FFT Structure for N=8

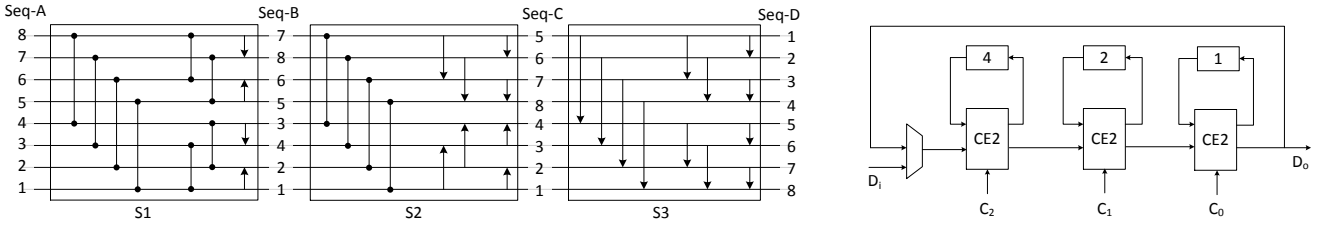
butterfly input pair for that stage in the SFG. In R2SDF FFT, the inputs are given in serial manner. The numbers on the delay–lines in Fig. 2 indicates the number of complex samples it can store.

The latency of the R2SDF structure is $\Theta(N)$ cycles that corresponds to the sum of all delay elements across all the stages. Since, this architecture is completely pipelined, a throughput of 1 clock cycle per sample can be achieved after the initial latency. Another advantage with R2SDF architecture is the simple control structure for FFT computation. The control structure requires a $\log_2 N$ bit binary counter [4][5].

In the Fig. 2, BF2 denotes a Radix-2 butterfly unit. This unit carries out addition, subtraction and twiddle factor multiplications. When the mux control (Cn) is 0, the butterfly remains idle and data passes through without any processing and when the control is 1, the butterfly unit processes the incoming samples. Every clock cycle, one of the butterfly outputs is stored back into the delay (or feedback) element and the other is passed onto the next stage.

IV. Proposed Dual-Purpose Hardware Accelerator Architecture

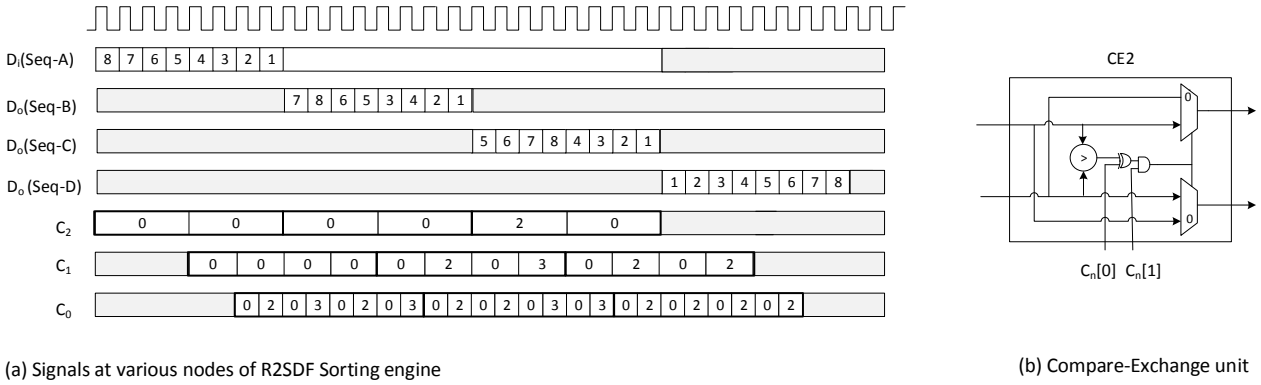
The similarity of signal permutation and spatial regularity of the bitonic sorting network with a FFT's SFG allow us to re-use R2SDF FFT engine for implementing the sorting algorithm. In this section, we present dual purpose hardware accelerator architecture. We describe both the bitonic sorting network and the R2SDF architecture in details in the below sections.



(a) Alternate SFG for a Bitonic network for N=8

(b) R2SDF implementation of Bitonic network for N=8

Fig.3 : Mapping a bitonic sorting network to a R2SDF structure for N = 8 (with number of iterations = 3)



(a) Signals at various nodes of R2SDF Sorting engine

(b) Compare-Exchange unit

Fig.4 : Waveforms for R2SDF Sorting Engine for N=8

A. Mapping the Bitonic Network to R2SDF structure

The proposed hardware sorting accelerator is based on the bitonic sorting algorithm implemented using the R2SDF structure. The bitonic network shown in Fig.1 is redrawn in Fig.3a to map it to an equivalent N point FFT SFG. New connectors (without an arrow) can be seen in this SFG that corresponds to "flow-through" operations, i.e. the data elements at the two ends of the connector do not perform a compare-exchange operation but rather flows through the network without any swap of data locations. The new SFG for sorting with "flow-through" enables mapping of each stage of the bitonic network to an N-point R2SDF. These "flow-through" connections in the SFG are introduced to maintain a steady flow of data across all the pipelined stages of the hardware engine similar to FFT data flow.

Fig.3-b shows the proposed hardware pipelined implementation of the bitonic sorting network for N=8 using the R2SDF FFT structure by replacing the Radix-2 butterfly (BF2) units with a 2-input compare-exchange units (CE2) as shown in Fig.4b. The R2SDF structure (with BF2 replaced with CE2) in Fig. 3b now maps to each stage of the bitonic network in Fig. 3a. We will call this structure an N-sample Bitonic Sorter (BS).The N-sample BS is used iteratively $\log_2 N$ times to implement an N-sample Bitonic Network.

Memory or shift registers in feedback path are used to implement the delay elements in all stages. The width of the delay elements must be equal to or greater than the width of the incoming data stream to be sorted. Hence, if the FFT uses delay elements of width b to store its intermediate data with real and imaginary parts of b/2 bits each, then the same delay elements can be re-used for sorting real data samples of bit-width b. The "distance" of the compare-exchange units in each stage is same as the "distance" of the butterfly units used in FFT pipeline stages. However, the "direction" of compare-exchange operations is not fixed during the course of the bitonic sort algorithm. This unit will thus have a 2-bit control to additionally specify the "direction" of compare-exchange as described below:

0: Bypass compare-exchange corresponding to "flow-through" operation in SFG. Store the data from the previous stage into its memory. Send the oldest data from its memory to the next stage

2: Compare data from previous stage with the oldest data in its memory. Store the larger data to its memory and pass the smaller data to the next stage.

3: Compare data from previous stage to the oldest data in its memory. Store the smaller data into its memory and pass the bigger data to the next stage.

This control code has a symmetry and pattern associated to it. This can be easily generated using counter bits from modulo-N binary counter and modulo-$\log_2 N$ binary counter associated with each stage.

For sorting a real data array of length N, all the $\log_2 N$ stages of the R2SDF structure (Bitonic Sorter-BS) are used with an extra feedback connection from the output back to the input. The feedback connection ensures iterative use of the R2SDF BS hardware $\log_2 N$ times, for implementing $\log_2 N$ stages of the bitonic network. In the first iteration, corresponding to the first stage of the bitonic sorting network, the inputs are directly fed into the sorting engine and only last pipeline stage (butterfly distance-1) is engaged for compare-exchange operation. All the other stages operate in bypass mode for the CE2 unit. In the second iteration,

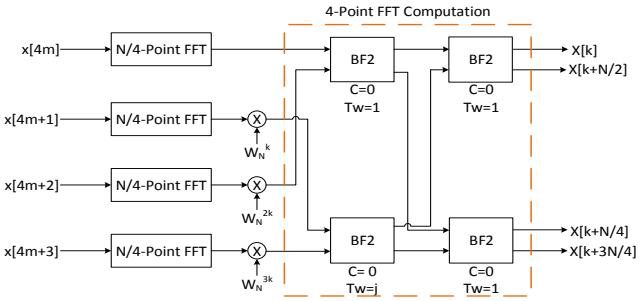


Fig. 5 : 4x Throughput R2SDF FFT Accelerator

corresponding to the second stage of the sorting network, the last two pipeline stages (distance-2 and distance-1) are engaged for compare-exchange operations with the remaining stages in bypass mode and so on. In the final iteration, compare-exchange units of all the $\log_2 N$ stages are engaged. Irrespective of the number of compare-exchange units engaged in each iteration, all the $\log_2 N$ iterations have a fixed latency of $\Theta(N)$ cycles. This is equal to the inherent latency built into the R2SDF FFT structure. Hence, total latency of the sorting engine using the R2SDF architecture is equal to $\Theta(N*\log_2 N)$ clock cycles. However, the R2SDF structure when re-used in the context of sorting is not truly pipelined since the hardware is locked until the $\log_2 N$ iterations are completed. Hence, the hardware can only sort N elements in $\Theta(N*\log_2 N)$ clock cycles before it can take the next sample set for sorting. Thus, the effective throughput of the R2SDF sorting accelerator is $\Theta(\log_2 N)$ clock cycles per sample

Fig.4a shows a sample data flow at the input and output nodes of a sorting accelerator for N=8 with random input data pattern. The entire sorting operation takes $N*\log_2 N$ clock cycles, i.e. 24 clock cycles to complete from the time last data was fed to the sorting accelerator.

V. Improved Parallelism for Combined Sorting and FFT Harware Accelerator

The FFT and sorting hardware accelerators described in the previous sections are serial in nature i.e. they take in serial data and gives out serial data after a fixed latency. For a high performance system where the processor, bus fabric and DMA has a wider bandwidth and can handle a higher throughput, the relatively slow rate of data consumption and data generation by the hardware accelerators brings down the overall system performance. Hence, to avoid being a performance bottleneck for the system, there is a need to improve the data parallelism of the combined FFT and Sorting Hardware Accelerator.

The FFT throughput can be increased by using higher radix FFT structures. However, such higher radix FFTs are typically very complex to design and the hardware re-use between FFT engine and sorting engine also becomes very challenging. Therefore, we have implemented a high throughput FFT accelerator by stitching 4 parallel R2SDF FFT engines in parallel as shown in Fig. 5. Here, m is used to denote the sample indices in time-domain and k to denote the indices in frequency domain where m and n varies from 0 to (N/4)-1. In this architecture, 4 consecutive time domain samples are fed to the 4 N/4 R2SDF FFT engines in parallel. The four outputs of the N/4 point FFT engines are then stitched together using a single radix-4 butterfly without incurring any extra latency. In addition, both control logic

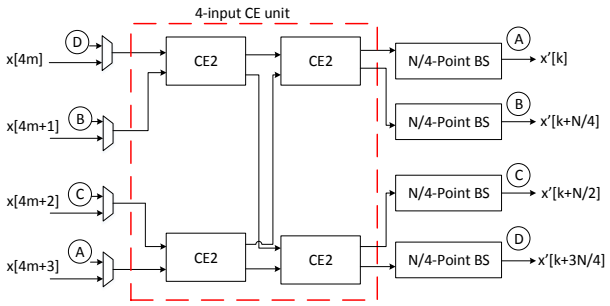


Fig. 6 : 4x Throughput R2SDF Sorting Accelerator

and twiddle factors are shared across all the 4 N/4 FFT engines since they perform the same operation every clock. The latency of the proposed FFT accelerator is thus reduced to N/4 clock cycles, corresponding to the latency of the N/4 point FFT engine with a 4 times increase in throughput. The memory complexity of the 4x throughput FFT architecture still remains $\Theta(N)$. However, the number of butterfly units (BF2) increases in number by a factor of 4.

The same hardware structure can be easily reused by the sorting engine to improve its throughput and latency numbers by 4x as shown in Fig. 6. Here, m is used to represent the sample indices for input data and k used to represent the output data ,where m and n varies from 0 to (N/4)-1 .The number of iterations for sorting a data array of length N still remains $\log_2 N$. However, the clock cycles required for each iteration is reduced by a factor of 4 due to 4x parallelism introduced. The latency of the 4x sorting accelerator is thus, $\Theta((N*\log_2 N)/4)$ clock cycles with an effective throughput of Θ $((\log_2 N)/4)$ clock cycles per sample.A 4-input compare-exchange(CE) unit is introduced to perform the permutation of output data samples streamed out from the 4 parallel N/4 sorting engines. This 4-input CE unit is engaged for compare-exchange operation only during the last 2 iterations and it remains in bypass mode for the initial iterations. The outputs of the 4 parallel N/4 sorting engines(BS) are looped back to the input of the 4-input compare-exchange units $(\log_2 N)$-1 times.

VI. Results

The area of a dual-purpose hardware accelerator (1x - serial) that implements an FFT with complex data bit-width of 24bits and a Sorting engine with real-data bit width of 48bits, synthesized at 400MHz is 0.44 sq.mm in 45nm CMOS technology for N=4096.The area numbers for the standard cell logic are largely dominated by the complex multipliers in butterfly units for FFT. The delay elements are implemented as single-port RAMs.

For a 4x throughput implementation, the area number is 0.98 sq.mm. This increase in area is largely contributed due to a 4x increase in the number of butterfly units. The additional area needed for implementing a sorting engine for a 4x implementation with N=4096 and 48bit real data input involves 44 binary comparators and 44*4*48 2:1 multiplexors and some additional control logic. This logic turns out to be less than 0.06 sq.mm, that is around 5% of the total accelerator area. The area and speed of the dual-purpose FFT and sorting accelerator is summarized in Table 1

TABLE 1: Dual Purpose Accelerator Area and Speed

Accelerator Throughput Mode	Processing Engine	Area (Sq.mm)	Cycles@400Mhz (N = 4096)
1x	FFT Engine	0.44	4108
	Sorting Engine		49296
4x	FFT Engine	0.98	1034
	Sorting Engine		12408

VII. CONCLUSIONS

We have shown that a standard hardware accelerator used to compute an N point FFT using an R2SDF architecture can be re-used as a sorting accelerator without any significant penalty in the circuit area. We first presented a novel serial sorting implementation with time and memory complexity of $\Theta(N*\log_2 N)$ and $\Theta(N)$ that fits into a N-point FFT hardware structure. We have further proposed a common architecture with increased parallelism that results in a 4x improvement in the throughput of both the FFT and the sorting accelerators without any increase in memory complexity. The proposed dual-purpose hardware accelerator architecture can also be used to implement a standalone FFT or sorting accelerator based on the system integration requirement.

REFERENCES

[1] D. E. Knuth, The Art of Computer Programming: Sorting and Searching, Reading, MA: Addison-Wesley, 1998.

[2] T. Cormen, C. L. R. Rivest, and S. Clifford, Introduction to Algorithms, 2nd ed. Boston, MA: McGraw-Hill Book Company, 2001

[3] K. E. Batcher, "Sorting networks and their applications," in Proc. AFIPS Spring Joint Comput. Conf., vol. 32, pp. 307–314, 1968.

[4] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation", IEEE Transactions on Computers, vol. C-33, no. 5, pp. 414-426, May 1984.

[5] S. He and M. Torkelson, "A new approach to pipeline FFT processor," in Proc. IEEE Int. Parallel Processing Symp., 1996, pp. 766–770