SERIAL AND INTERLEAVED ARCHITECTURES FOR COMPUTING REAL FFT

Aravinth Chinnapalanichamy and Keshab K. Parhi, *Fellow, IEEE*
Department of Electrical and Computer Engineering
University of Minnesota, Twin cities
Minneapolis, MN, USA
{chinn028, parhi}@umn.edu

ABSTRACT

The Fast Fourier transform (FFT) is an important operation in digital signal processing applications. In applications such as biomedical signal processing, the signals are real. The real-valued signals exhibit conjugate symmetry, giving rise to redundant values in the outputs. This property can be exploited to reduce arithmetic computations, area and power consumption. This paper presents hardware architectures for computing *real* FFT that exploits this conjugate symmetry property where the inputs are processed in a *serial* manner. This is facilitated by pushing the twiddle factor values across various butterfly stages. In this paper, two different serial FFT architectures are presented: one using *real* and the other using *hybrid* datapaths. These architectures process one sample per clock cycle and are well suited for low-sample-rate applications such as biomedical. These architectures are also modified so that two independent computations can be *interleaved* in the same datapath. The advantage of interleaving is reduction in area, and is attractive for applications where FFT computation of two independent real signals is required.

Index Terms **–** FFT, Real FFT, Serial Processing, Interleaving, Real Datapath, Hybrid Datapath

1. INTRODUCTION

Fast Fourier transform is used extensively in many signal processing applications. FFT architectures are based on the most widely used Cooley-Tukey algorithm [1]; this is a divide and conquer algorithm that recursively transforms a DFT of size N = *N1.N2* into two smaller DFTs of sizes N1 and N2, along with O (N) multiplications referred to as twiddle factors. In general, FFT engines operate over complex inputs and the butterflies used in these architectures also process complex samples.

There has been an increasing interest in FFT computation for *real* valued signals in recent years. Many of the biomedical signals such as electrocardiogram (ECG) and electroencephalogram (EEG) are real. Several architectures have been recently proposed for computing real FFT. In these architectures, nearly half the computations are eliminated. The RFFT architectures [2]-[7] use either fully-real or hybrid data paths. A real butterfly accepts only two real inputs as opposed to a complex butterfly that processes four real values corresponding to two complex inputs. Although a real datapath based architecture minimizes the number of samples that are computed at the end of each FFT stage, these architectures require more area and power compared to architectures based on hybrid data paths. Parallel real FFT architectures using real and hybrid data paths have been presented in [7] and [2], respectively. These architectures process multiple consecutive samples every clock cycle, and achieve full hardware utilization with respect to the butterflies as well as the delay elements.

The main objective of this paper is design of serial real FFT architectures where area is a main constraint and sample rates required are very low. Examples of these applications include processing of ECG, EEG, or heart sound. Typically sampling rates in these applications rarely exceed 1 kHz. Often processing of signals from multiple channels or electrodes is of interest. Serial architectures for computing complex FFT have been presented in [8]; however, no architecture has been presented for computing real FFT in a serial manner where one sample is processed every cycle. In these architectures, the butterflies are half-utilized; however, the delay elements are fully utilized. Serial architectures for computing real FFT of two independent channels are also presented using both real and hybrid datapaths. It is shown that the area overhead for interleaving two channels is less than 20% compared to without interleaving.

The rest of this paper is organized as follows. Section 2 reviews different FFT architectures implemented based on Cooley-Tukey algorithm. The proposed serial RFFT implementations are presented in Section 3. Section 4 presents comparisons of hardware area and power consumption of the proposed serial architectures. The conclusions are presented in Section 5.

2. REVIEW OF FFT ARCHITECTURES FOR REAL INPUTS

2.1. L- Parallel FFT

The level of parallelism determines the number of inputs that a single stage can process in a single cycle. In most cases, it is a design choice that one makes while formulating the folding factor for the FFT flow graph. Sometimes, this leads to underutilization of hardware

components which can be seen in [3] where the folding set used to design the hardware contains null operations. However, as described in [3], the folding set can be modified such that the null operations can be replaced with useful calculations.

2.2. Radix 2,2^2 & 2^3 FFT

The radix values used for designing an FFT architecture can play a significant role in reducing the hardware. The number of complex multipliers required depends on the radix chosen. For example, if the number of multipliers required in a radix-2 implementation is N, then the number of complex multipliers required for radix-2^2 and

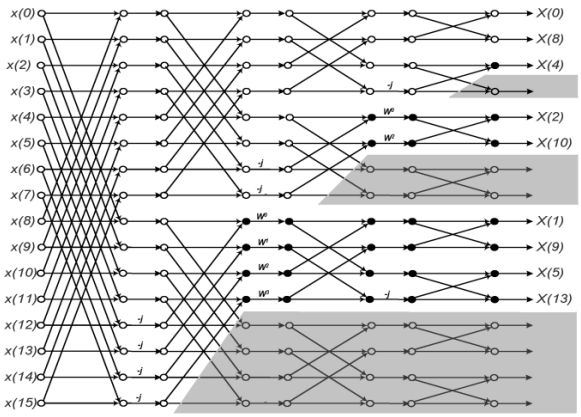

Fig. 1. DIF Flow graph

radix-2^3 are N/2 and N/4, respectively. The area benefits for higher radices are significantly higher compared to lower radices.

2.3. Datapath style – Fully-Real, Fully-Complex and Hybrid

The datapaths chosen for FFT architectures also have impact on area and power consumption. For real inputs, a complex/hybrid datapath leads to a large area penalty; however, the latency of these architectures is less than that of a fully-real datapath [7]. In addition, there has to be a real-imaginary combiner circuit at the output for a fully-real datapath.

Another major disadvantage with fully-real datapath is that it requires more complex multipliers whereas in complex and hybrid datapaths, there is more flexibility in pushing the twiddle factors across the butterfly stages and replace complex multipliers with simple multipliers.

3. PROPOSED RADIX-2 SERIAL FFT ARCHITECTURES

The flow graph, shown in Fig. 1, represents decimation-in-frequency (DIF) FFT computation for real inputs. The unnecessary computations for real inputs, as highlighted in the figure, are avoided by removing their calculations from the datapath. Each line in the flow graph represents two physical wires, one for carrying the real part of a computation value and the other for the imaginary part. However, when the inputs are real, certain lines in the flow graph can use a single wire, i.e., either real or imaginary part of a value. Complex values are generated only when a real value passes through a complex multiplier. This can be seen in Fig. 1 where lines between bold dots represent complex data values and normal lines represent real values. Thus, when folding, instead of placing null operations, useful computations can be carried out by splitting real and imaginary parts of a complex value and expanding the imaginary part to fill the removed part. Since real and imaginary parts of a complex value traveling along different routes have to be scaled by the same twiddle factor value after each butterfly stage, additional delay elements as described in [7] are required as part of the control circuit.

3.1. Fully-Real Serial FFT Datapath

The proposed fully-real RFFT structure is shown in Fig. 2. The feedback in the architecture reduces the number of delays and improves the hardware utilization efficiency of the delays to 100%, similar to the serial complex FFT architectures. For an N point RFFT, the block marked in Fig. 2 gets repeated. The last stage multiplier requires only multiplication by W^2 which can be computed using simple addition and scaling operations. A canonical signed digit multiplier [9] can be used for this purpose. A butterfly, at any cycle, produces two outputs and one of them is fed back to the delay elements, so that a serial data-flow is realized. The data that is sent back to the delay elements later passes through the butterfly without any operations. For this purpose, a provision for directly bypassing the input to the output (bypassing) using a multiplexer structure is incorporated into the butterfly as shown in Fig. 3. The same bypassing is also provided for the multiplier block.

In order to synchronize data in accordance with the timing, multiplexers, demultiplexers, switches and delay elements are used. Two types of de-multiplexers (D1 & D2), multiplexers (M1 & M2) and switches (S1 & S2) are used as shown in Fig. 4. Control signals for these blocks are shown in Table I. For an N-point RFFT with *n* stages, all the control signals can be generated using a single *n*-bit counter with simple combinational logic. For each stage of the RFFT containing M delay elements, the control signals follow a period of 2M and 4M.

The following folding set is used for a 16 point Fully-Real FFT design with 1 stage pipelining between stages.

A={ ϕ,ϕ, ϕ,ϕ ,ϕ,ϕ, ϕ,ϕ, A0, A1, A2, A3, A4, A5, A6, A7}

B={B3,ϕ,ϕ,ϕ,ϕ,B4_p,B5_p,B6_p,B7_p,ϕ,ϕ,ϕ,ϕ,B0,B1,B2}

C={C0,C1,ϕ,ϕ,C2_p, c3_p,ϕ,ϕ, C4, C5, ϕ,ϕ, C6, C7,ϕ, ϕ}

D={ϕ,D6,ϕ,D7,ϕ,D0,ϕ, D1_p, ϕ, D2, ϕ, D3, ϕ, D4, ϕ, D5}

In the folding set, A, B, C and D represent butterfly operations at stages 1, 2, 3, 4, respectively. The passing operations used for real-imaginary splitting are denoted as

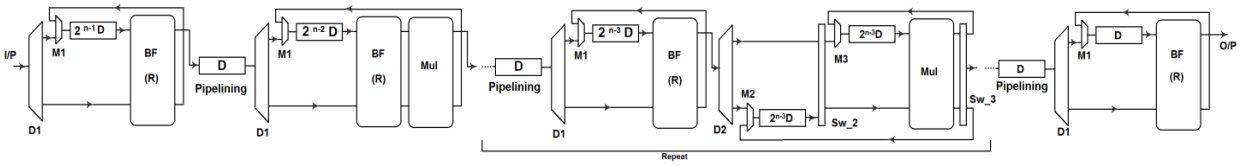

Fig. 2. Fully-real serial RFFT architecture

TABLE I. CONTROL SIGNALS FOR THE SERIAL ARCHITECTURES.

Control unit	Period	Control signal
D1,M1	2M	M cycles - 0 M cycles - 1
D2,M2	4M	M cycles - 1 3M cycles - 0
Sw_2, M3	4M	M cycles - 1 2M cycles - 0 M cycles - 1
Sw_3 – C1	4M	M cycles - 0 M cycles -1 2M cycles - 0
Sw_3 – C2	4M	M cycles - 1 3M cycles - 0

Bi_p, Ci_p and Di_p. The timing diagram for this 16-point fully-real implementation is shown in Fig. 5.

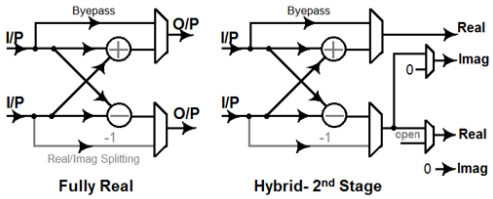

Fig. 3. Butterfly modules for fully-real and hybrid structures.

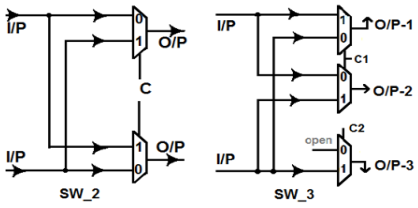

Fig. 4. Switches

To process multiple channels using the same hardware, an interleaved FFT architecture is proposed. However, to facilitate interleaving, additional delay elements are required. This interleaving operation can be implemented using a simple multiplexer as shown in Fig. 6. The control signals are made K-slow for a K-way interleaving operation [9].

3.2. Hybrid Datapath

The proposed hybrid serial architecture is shown in Fig. 7. Here only the first two stage butterflies are real, and the rest are complex. At those stages, where complex

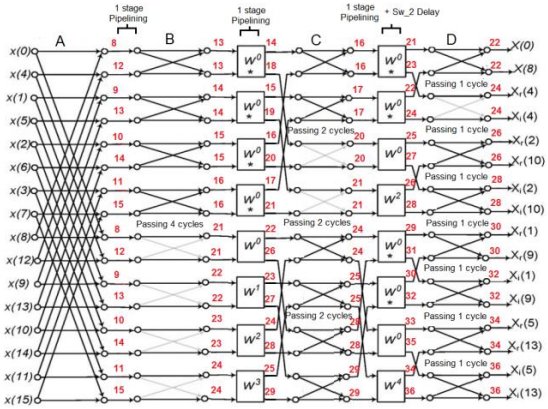

Fig. 5. Flow-graph with scheduling for 16-point fully-real implementation

butterflies are used, a single line in the dataflow graph corresponds to two physical wires, as shown by bold lines in Fig. 7. This also doubles the number of delay elements as compared to the real FFT architecture. The 2nd butterfly block in the hybrid datapath is shown in Fig. 3. It can be seen in the figure that there is a provision for combining the two inputs to the butterfly as a single complex value. This facility is required as the rest of the datapath is complex. The complex butterflies used in this hybrid design is a traditional one with provision for bypassing the input values.

The control signals for both architectures are periodic with period 2M cycles which can be generated using a $\log_2$N-bit counter. One of the main advantages of the hybrid architecture is that the feedback loop does not contain the multiplier block. Hence, it enables pipelining of the multiplier circuit as demanded by the application. Similar to the serial architecture, the hardware in the hybrid structure is also underutilized due to the feedback structure which requires simple bypassing operation across the butterfly block during which the butterfly block is not utilized. The following folding set is used for a 16 point Hybrid FFT implementation with 2-stage pipelining between stages.

A={ϕ, ϕ,ϕ, ϕ,ϕ, ϕ, ϕ, ϕ, A0, A1, A2, A3, A4, A5, A6, A7}

B={B3,ϕ,ϕ,ϕ,ϕ,B4_c,B5_c,B6_c,B7_c,ϕ,ϕ,ϕ,ϕ,B0,B1,B2}

C={ϕ,C0,C1,ϕ,ϕ,C2_c,c3_c, ϕ, ϕ, C4, C5, ϕ, ϕ, ϕ, ϕ, ϕ, ϕ }

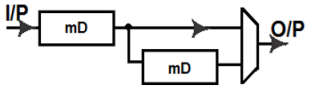

Fig. 6. Control structure for interleaving by 2

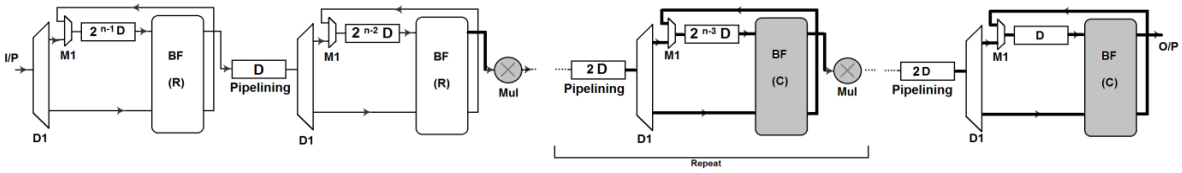

Fig. 7. Hybrid serial RFFT architecture

D={ϕ, ϕ, ϕ, ϕ, D0, ϕ, D1_c, ϕ, D2, ϕ, ϕ, ϕ, D4, ϕ, D5, ϕ}

In the folding set, A, B, C and D represent butterfly operations at stages 1, 2, 3, 4, respectively. Input combining operations are denoted as Bi_c, Ci_c and Di_c. The timing diagram for this 16 point hybrid implementation is shown in Fig. 8.

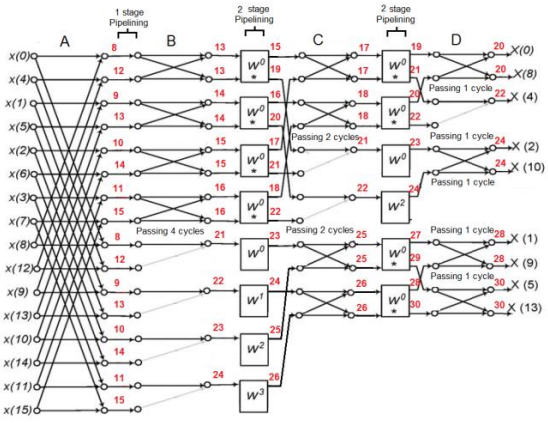

Fig. 8 Flow-graph with scheduling for 16-point hybrid implementation

4. PERFORMANCE COMPARISON

In terms of performance, the hybrid architecture is superior compared to a fully-real design, as the multiplier is not part of the feedback loop. In addition, the latency of this architecture is approximately three-fourth compared to a fully-real design.

TABLE II. HARDWARE COST COMPARISON OF SERIAL RFFT

Architecture	Complex Multiplier	Real adder	Delays
Hybrid	Log_2N - 3	$4log_2N$ – 4	5N/4-2
Hybrid- Interleaved by factor K	Log_2N - 3	$4log_2N$ - 4	K(5N/4-2)
Fully-real	Log_2N - 3	$2log_2N$	3N/2 - 5
Full- real- Interleaved by factor K	Log_2N - 3	$2log_2N$	K(3N/2- 5)

Table II compares the hardware cost of the four architectures presented in this paper. These include: real/hybrid architectures and serial/interleaved. Table II shows that the real and hybrid architectures require the same number of multipliers; however, the hybrid architectures require more adders. Furthermore, the additional cost of interleaving is the increase in the number of delay elements by 100%.

Table III presents the synthesis results obtained for the proposed architectures. The four designs were synthesized using a clock speed of 100 MHZ in Synopsys Design Compiler. It can be seen that the hybrid datapath requires about 20% increase in area and about 21% increase in power compared to the real datapath. Although the hybrid design requires less delays than real design, it has more area. This may be attributed to the cost of additional adders and interconnections since the inter-stage communicating signals are complex. It may also be attributed to the sub-optimal optimization by the Design Compiler. The area increase due to interleaving is about 18% for hybrid and 30% for real design. This is significantly less than just duplicating the serial datapath twice. The hybrid datapath was also optimized for speed. The highest speed achieved for the hybrid datapath is with no internal pipelining of the multiplier is 437 MHz whereas that for the real datapath is 100 MHz.

TABLE III. SYNTHESIS RESULTS OF 1024-POINT SERIAL RFFT USING 45NM NCSU PDK FOR 100 MHZ CLOCK FREQUENCY

Architecture	Area	Power
Hybrid	0.346152 mm2	17.165 mW
Hybrid- Interleaved by factor 2	0.410346 mm2	19.153 mW
Fully-real	0.284327 mm2	14.8012 mW
Fully-real- Interleaved by factor 2	0.375221 mm2	17.7314 mW

5. CONCLUSION

This paper has presented four different serial architectures for radix-2 computation of RFFT; these include real/hybrid and serial/interleaved. It can be seen that a hybrid version of the serial architecture has the main advantage that the multiplier is outside the feedback loop. Thus, this architecture can be fine-grain pipelined and can be operated with very low supply voltage for low-power applications. The interleaved designs are attractive for biomedical applications. Interleaving avoids doubling the area due to replication of datapath. Power spectral density (PSD) is an important feature for classification of biomedical signals [10]. Future work will be directed towards computing PSD of the real signals using the FFT architectures presented in this paper.

6. ACKNOWLEDGMENT

The authors thank Mr. Vishal Vijayakumar and Mr. Amey Naik for numerous discussions on the control circuit for interleaved FFTs.

7. REFERENCES

[1] A. V. Oppenheim, R. W. Schafer, *Discrete-Time Signal Processing*, (3rd Edition), Prentice Hall Signal Processing, 2009

[2] M. Ayinala and K.K. Parhi, "FFT Architectures for Real-valued Signals based on Radix-2^3 and Radix-2^4 algorithms," *IEEE Trans. Circuits and Systems-I: Regular Papers*, 60(9), pp. 2422-2430, Sept. 2013

[3] M. Ayinala, M.J. Brown and K.K. Parhi, "Pipelined Parallel FFT Architectures via Folding Transformation", *IEEE Trans. VLSI Systems*, pp. 1068-1081, 20(6), June 2012

[4] H. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 35, no. 6, pp. 849–863, Jun 1987.

[5] M. Garrido, K. K. Parhi, and J. Grajal, "A pipelined FFT architecture for real-valued signals," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 12, pp. 2634–2643, Dec. 2009.

[6] Y. Voronenko and M. Püschel, "Algebraic signal processing theory: Cooley-Tukey type algorithms for real DFTs," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 205–222, Jan. 2009.

[7] S.A. Salehi, R.Amirfattahi, and K.K. Parhi, "Pipelined Architectures for Real-Valued FFT and Hermitian-Symmetric IFFT with Real Datapaths," *IEEE Trans. Circuits and Systems-II: Transactions Briefs*, 60(8), pp. 507-511, Aug. 2013

[8] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integr. Circuits Conf.*, Santa Clara, CA, USA, May 1998, pp. 131–134.

[9] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Hoboken, NJ, USA: Wiley, 1999.

[10] K.K. Parhi and M. Ayinala, "Low-Complexity Welch Power Spectral Density Computation," *IEEE Trans. Circuits and Systems-I: Regular Papers*, 61(1), pp. 172-182, Jan. 2014