Architectures for Dynamic Data Scaling in 2/4/8K Pipeline FFT Cores

Lenart, Thomas; Öwall, Viktor

Published in:
IEEE Transactions on Very Large Scale Integration (VLSI) Systems

DOI:
10.1109/TVLSI.2006.886407

2006

[3] D. Garrett, L. Davis, S. Brink, B. Hochwald, and G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE J. Solid-State Circuits*, vol. 39, no. 9, pp. 1544–1552, Sep. 2004.
[4] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. ISSSE*, 1998, pp. 295–300.
[5] J. Wang and B. Daneshrad, "A comparative study of MIMO detection algorithms for wideband spatial multiplexing systems," *IEEE Wireless Commun. Netw. Conf.*, vol. 1, no. 2, pp. 408–413, Mar. 2005.
[6] G. H. Golub and C. F. Van Loan*, Matrix Computation*. Hanover, PA: Johns Hopkins Univ. Press, 1996.
[7] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2000, pp. II737–II740.
[8] Z. Guo and P. Nilsson, "A VLSI implementation of MIMO detection for future wireless communications," in *Proc. IEEE 14th Personal, Indoor Mobile Radio Commun.*, 2003, pp. 29–49.
[9] Z. Khan, T. Arslan, J. S. Thompson, and A. T. Erdogan, "Dual strategy based VLSI architecture for computing pseudo inverse of channel matrix in a MIMO wireless system," in *Proc. IEEE Int. Symp. VLSI*, 2006, pp. 12–17.
[10] N. D. Hemkumar and J. R. Cavallaro, "A systolic VLSI architecture for complex SVD," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1992, pp. 1061–1064.
[11] C. M. Rader, "VLSI systolic arrays for adaptive nulling," *IEEE Signal Process. Mag.*, vol. 13, no. 4, pp. 29–49, Jul. 1996.

Architectures for Dynamic Data Scaling in 2/4/8K Pipeline FFT Cores

Thomas Lenart and Viktor Öwall

***Abstract*—This paper presents architectures for supporting dynamic data scaling in pipeline fast Fourier transforms (FFTs), suitable when implementing large size FFTs in applications such as digital video broadcasting and digital holographic imaging. In a pipeline FFT, data is continuously streaming and must, hence, be scaled without stalling the dataflow. We propose a hybrid floating-point scheme with tailored exponent datapath, and a co-optimized architecture between hybrid floating point and block floating point (BFP) to reduce memory requirements for 2-D signal processing. The presented co-optimization generates a higher signal-to-quantization-noise ratio and requires less memory than for instance convergent BFP. A 2048-point pipeline FFT has been fabricated in a standard-CMOS process from AMI Semiconductor (Lenart and Öwall, 2003), and a field-programmable gate array prototype integrating a 2-D FFT core in a larger design shows that the architecture is suitable for image reconstruction in digital holographic imaging.**

***Index Terms*—Block floating point (BFP), convergent BFP (CBFP), digital holography, digital video broadcasting (DVB), dynamic data scaling, fast Fourier transform (FFT), hybrid floating point, orthogonal frequency-division multiplexing (OFDM).**

I. Introduction

The fast Fourier transform (FFT), is one of the most commonly used operations in digital signal processing and, currently, the demands increase towards larger and multidimensional transforms. Larger transforms require more processing on each data sample, which increases the total quantization noise. This can be avoided by gradually increasing the wordlength inside the pipeline, but affects memory requirements as well as the critical path in arithmetic components. For large size FFTs, dynamic scaling is, therefore, a suitable tradeoff between arithmetic complexity and memory requirements. The following architectures have been evaluated and compared with related work.

a) A hybrid floating-point pipeline with fixed-point input and tailored exponent datapath for 1-D FFT computation.
b) A hybrid floating-point pipeline for 2-D FFT computation, which also requires the input format to be hybrid floating point. Hence, the hardware cost is slightly higher than in (a).
c) A co-optimized design based on a hybrid floating-point pipeline combined with block floating point (BFP) for 2-D FFT computation. This architecture has the processing abilities of (b) with hardware requirements comparable to (a).

The primary target application for the implemented FFT core is a microscope based on digital holography [2] where visible images are to be digitally reconstructed from an interference pattern. The pattern is recorded on a large digital image sensor with a resolution of 2048 × 2048 pixels and processed by a reconstruction algorithm based on a 2-D Fourier transformation. Hence, the architectures outlined in (b) and (c) are suitable for this application. Another area of interest is in wireless communication systems based on orthogonal frequency division multiplexing (OFDM). The OFDM scheme is used in, for example, digital video broadcasting (DVB) [3], including DVB-T with 2/8-K FFT modes and DVB-H with an additional 4-K FFT mode. The architecture described in (a) is suitable for this field of application.

Section II gives a brief introduction to the FFT and Section III presents different dynamic data scaling alternatives for pipeline FFTs, with additional architectural features described in Section IV. Section V shows software simulation results in terms of precision and memory requirements. Finally, Section VI presents the VLSI implementation and measurements on the fabricated application-specific integrated circuit (ASIC) prototype, and a conclusion is given in Section VII.

II. FFT Architecture

The fast Fourier transform is a decomposition of an N-point discrete Fourier transform (DFT) into successively smaller DFT transforms [4]. This paper describes *pipeline* FFT architectures, constructed from a number of cascaded Radix-r butterfly blocks and complex multipliers each dividing the sequence into r smaller FFTs. Another common approach is *parallel* FFT architectures, placing the computational blocks in parallel instead of cascaded. Simulations and implementations presented in this paper are all based on the Radix-2^2 single path delay feedback $(R2^2\mathrm{SDF})$ algorithm [5]. Input data is supplied in linear sample order, hence, requiring the largest delay feedback buffer of size $N_{\mathrm{FFT}}/2$ in the initial butterfly unit. The N_{FFT}-point Radix-2^2 pipeline has low memory requirements ($N_{\mathrm{FFT}} - 1$ words), simple butterfly architecture and requires only $\log_4 N_{\mathrm{FFT}} - 1$ complex multipliers. The Radix-2^2 butterfly is constructed from two Radix-2 butterflies divided by a trivial multiplication, as shown in Fig. 1.

III. Dynamic Data Scaling

Fixed-point is a widely used format in realtime and low-power applications due to the simple implementation of arithmetic units. In fixed-point arithmetic, a result from a multiplication is usually rounded or truncated to avoid a significantly increased wordlength, hence, generating a quantization error. The quantization energy caused by rounding is relatively constant due to the fixed location of the binary

Manuscript received December 19, 2005; revised March 29, 2006.
The authors are with the Lund Institute of Technology, Department of Electroscience, SE-221 00 Lund, Sweden (e-mail: Thomas.lenart@es.lth.se).
Digital Object Identifier 10.1109/TVLSI.2006.886407

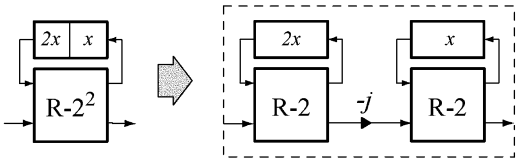

Fig. 1. Radix-2^2 ($R-2^2$) butterfly is constructed from two Radix-2 ($R-2$) butterflies, separated with a trivial multiplication.

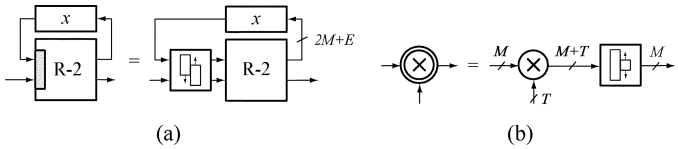

Fig. 2. Building blocks for a hybrid floating-point implementation. (a) Symbol of a modified butterfly containing an align unit on the input. (b) Symbol for a modified complex multiplier containing a normalization unit.

point, whereas the total energy depends on how the input signal utilizes the available dynamic range. Therefore, precision in the calculations depends on properties of the input signal, caused by uniform resolution over the total dynamic range. Fixed-point arithmetic usually requires an increased wordlength due to the tradeoff between dynamic range and precision. By using floating point and dynamically changing the quantization steps, the energy in the error signal will follow the energy in the input signal and the resulting signal-to-quantization-noise ratio (SQNR) will remain relatively constant over a large dynamic range. This is desirable to generate a high signal quality, less dependent on the transform length. However, floating point arithmetic is considerably more expensive in terms of chip area and power consumption, and alternatives will be presented in the following sections followed by a comparison in Section V.

A. Hybrid Floating-Point

Floating-point arithmetic increases the dynamic range by expressing numbers with a mantissa m and an exponent e, represented with M and E bits, respectively. A hybrid and simplified scheme for floating point representation of complex numbers is to use a single exponent for the real and imaginary part. Besides reduced complexity in the arithmetic units the total wordlength for a complex number is reduced from $2\times(M+E)$ to $2\times M+E$ bits. Supporting hybrid floating point requires pre- and post-processing units in the arithmetic building blocks, and Fig. 2 defines symbols used for representing these units. The FFT twiddle factors are represented with T bits.

B. BFP

BFP combines the advantages of simple fixed-point arithmetic with floating point dynamic range. A single exponent is assigned to a group of values to reduce memory requirements and arithmetic complexity. However, output signal quality depends on the block size and characteristics of the input signal [6]. Finding a common exponent requires processing of the complete block. This information is directly available in a *parallel* FFT architecture, but for *pipeline* FFT architectures scaling becomes more complicated since data is continuously streaming. A scheme known as convergent BFP (CBFP) has been proposed for pipeline architectures [7]. By placing buffers between intermediate stages, data can be rescaled using BFP, as shown in Fig. 3. The block size will decrease as data propagates through the pipeline until each value has its own exponent. Intermediate buffering of data between each stage requires a large amount of memory, and in practical applications the first intermediate buffer is often omitted to save storage space. However, this leads to a reduced SQNR as will

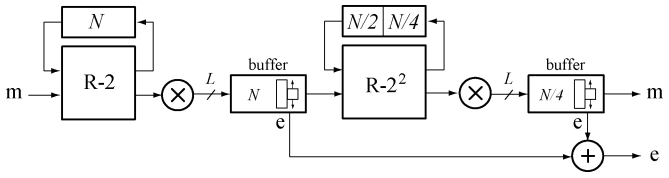

Fig. 3. Example of CBFP. The buffer after each complex multiplier selects a common exponent for a group of values, allowing fixed-point butterfly units. The first buffer is often omitted to save storage space, but will have a negative impact on signal quality.

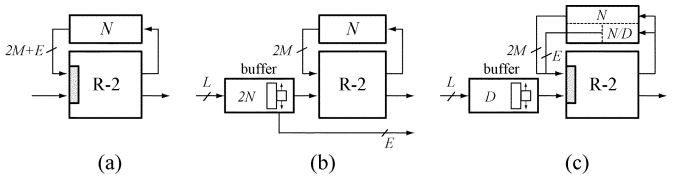

Fig. 4. (a) Hybrid floating point. (b) CBFP with $D=2N-1$ using a large buffer and fixed-point butterfly. (c) Small buffer reduces the exponent storage space in the delay feedback memory.

be shown in Section V and referred to as CBFP_{low} due to the lower memory requirements.

C. Co-Optimization

In this section, a co-optimized architecture that combines hybrid floating point and BFP is proposed. By extending the hybrid floating point architecture with small intermediate buffers, the size of the delay feedback memory can be reduced. Fig. 4(a)–(c) show dynamic data scaling for hybrid floating point, CBFP, and the proposed co-optimization architecture. Fig. 4(c) is a combined architecture with an intermediate buffer to apply block scaling on D elements, which reduces the storage space for exponents in the delay feedback memory with a factor D. We will derive an expression to find optimum values for the block size in each butterfly stage i to minimize the memory requirements for supporting dynamic scaling. The equations can be used for all configurations in Fig. 4(a)–(c) by specifying $D=1$ for hybrid floating point and $D=2N_i$ for CBFP. The length of the delay feedback memory, or first-in, first-out (FIFO), at stage i is

$$N_i=2^i,\qquad 0\le i\le i_{\max}=\log_2 N_{\text{FFT}}-1$$

and the number of exponent bits for the same stage is denoted E_i. The block size D spans from single elements to $2N_i$, which can be expressed as

$$D(\alpha_i)=2^{\alpha_i},\qquad 0\le\alpha_i\le i+1.$$

The total bits required for supporting dynamic scaling is the sum of exponent bits in the delay feedback unit and the total size of the intermediate buffer. This can be expressed as

$$\text{Mem}_i=\underbrace{E_i\left\lfloor\frac{\gamma N_i}{D(\alpha_i)}\right\rfloor}_{\text{delay feedback}}+\underbrace{L\left(D(\alpha_i)-1\right)}_{\text{buffer}}\qquad(1)$$

where

$$\gamma=\begin{cases}1, & \text{Radix-2}\\ 3/2, & \text{Radix}-2^2\end{cases}$$

and

$$L=\begin{cases}2M+E_i, & i=i_{\max}\\ 2(M+T), & 0\le i<i_{\max}.\end{cases}$$

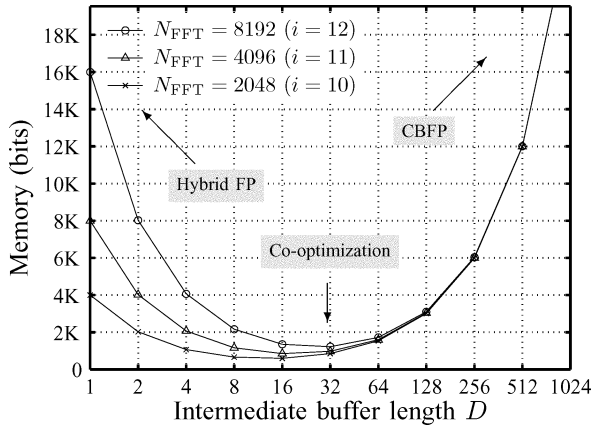

Fig. 5. Memory requirements for supporting dynamic scaling as a function of D for the initial butterfly in an $N_{\rm FFT}$ point FFT using data format $2 \times 10 + 4$. $D = 1$ represent a hybrid floating-point architecture, whereas $D \rightarrow N_{\rm FFT}$ approaches the CBFP architecture. Optimal value can be found in between these architectures.

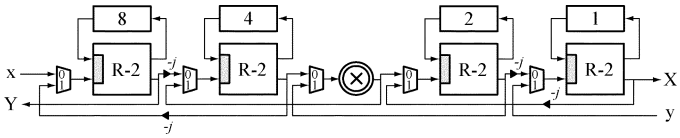

Fig. 6. Bidirectional 16-point FFT pipeline. The I/O to the left is in sequential order. The I/O to the right is in bit-reversed order.

For Radix-2^2 butterflies, (1) is only defined for odd values of i. This is compensated by a scale factor $\gamma = 3/2$ to include both delay feedback units in the Radix-2^2 butterfly, as shown in Fig. 1. The buffer input wordlength L differs between initial and internal butterflies. For every butterfly stage, α_i is chosen to minimize (1). For example, an 8192-point FFT using a hybrid floating-point format of $2 \times 10 + 4$ bits require 16 Kb of memory in the initial butterfly for storing exponents, as shown in Fig. 5. The number of memory elements for supporting dynamic scaling can be reduced to only 1256 bits by selecting a block size of 32, hence, removing over 90% of the storage space for exponents. The hardware overhead is a counter to keep track of when to update the block exponent in the delay feedback, similar to the exponent control logic required in CBFP implementations. Thus, the proposed co-optimization architecture supports hybrid floating point on the input port at very low hardware cost. Since the input and output format is the same, this architecture then becomes suitable for 2-D FFT computation.

IV. Architectural Extensions

The architectures described in this paper have been extended with support for bidirectional processing, which is important for the intended application and also in many general applications. A pipeline FFT can support a bidirectional dataflow if all internal butterfly stages have the same wordlength. The advantage with a bidirectional pipeline is that input data can be supplied either in linear or bit-reversed sample order by changing the dataflow direction. One application for the bidirectional pipeline is to exchange the FFT/IFFT structure using reordering buffers in an OFDM transceiver to minimize the required buffering for inserting and removing the cyclic suffix, proposed in [8]. OFDM implementations based on CBFP have also been proposed [9], but these solutions only operate in one direction since input and output format differ. Another application for a bidirectional pipeline is to evaluate 1-D and 2-D convolutions. Since the forward transform generates data in bit-reversed order, the architecture is more efficient if the inverse transform supports a bit-reversed input sequence as shown in Fig. 6. Both input and output from the convolution is in linear sample order, hence, no reorder buffers are required. The hardware requirement for a bidirectional pipeline is limited to multiplexers on the inputs of each butterfly and on each complex multiplier. Each unit requires 26 two-input muxes for internal $2 \times 11 + 4$ format, which is negligible compared to the size of an FFT stage.

TABLE I
Memory Requirements in Kb for Pipeline Architectures, Based on a 2048-Point Radix-2^2 With $M = 10$ and $E = 4$

Architecture	Delay feedback	Intermediate buffers	Total memory
1D Co-optimization	45.8	1.6	47.4
1D Hybrid FP (A)	49.0	-	49.0
1D CBFP_{low}	45.7	14.7	60.4
1D CBFP	45.7	60.0	105.7
2D Co-optimization(C)	50.0	0.4	50.4
2D Hybrid FP (B)	53.9	-	53.9

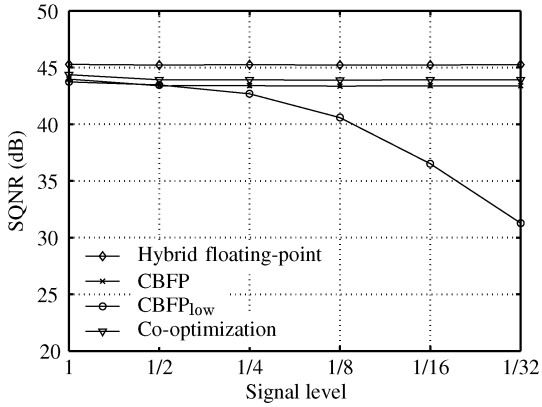

Fig. 7. Decreasing the energy in a random value input signal affects only the architecture when scaling is not applied in the initial stage. Signal level = 1 means utilizing the full dynamic range.

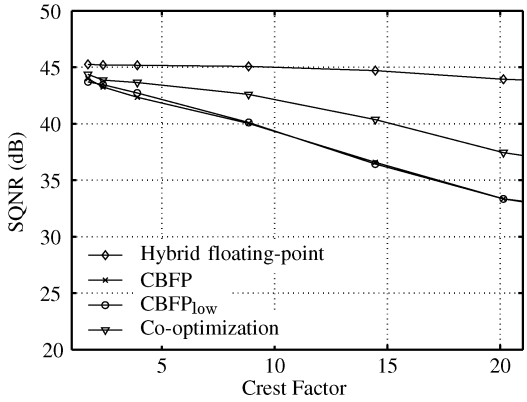

Fig. 8. Decreasing the energy in a random value input signal with peak values utilizing the full dynamic range. This affects all block scaling architectures, and the SQNR depends on the block size. The co-optimized architecture performs better than CBFP since it has a smaller block size through the pipeline.

V. Simulations

A simulation tool has been designed to evaluate different FFT architectures in terms of precision, dynamic range, memory requirements, and estimated chip size based on architectural descriptions. The user

TABLE II
COMPARISON BETWEEN PROPOSED ARCHITECTURES AND RELATED WORK (GREY FIELDS INDICATE ESTIMATED OR SIMULATED VALUES)

	Proposed (A)		Proposed (B)	Proposed (C)	Lin [10]	Bidet [7]	Wang [11]
Architecture	Pipeline (1D)		Pipeline (2D)	Pipeline (2D)	Parallel	Pipeline	Pipeline
Dynamic scaling	Hybrid FP		Hybrid FP	Co-optimization	BFP (D=64)	CBFP	No
Technology (μm)	0.35		Virtex-E	Virtex-E	0.18	0.5	0.35
Max Frequency (MHz)	76		50	50	56	22	16
Input wordlength	2×10		$2 \times 10 + 4$	$2 \times 10 + 4$	2×10	2×10	2×8
Internal wordlength	$2 \times 11 + (0 \ldots 4)$		$2 \times 11 + 4$	$2 \times 11 + 4$	$2 \times 11 + 4$	$2 \times 12 + 4$	$2 \times (19 \ldots 34)$
Transform size	2K	1/2/4/8K	2K	2K	8K	8K	2/8K
SQNR (dB)	45.3	44.0	45.3	44.3	41.2	42.4	
Memory (bits)	49K	196K	53.9K	50.4K	185K	350K	213K
Norm. Area (mm^2)[1]	7.58	≈ 16			18.3	49	33.75
Peak 1D Transform/s	37109	9277	24414	24414	3905	2686	7812/1953

[1] Area normalized to 0.35 μm technology.

can specify the number of bits for representing mantissa M, exponents E, twiddle factors T, FFT size (N_{FFT}), rounding type, and simulation stimuli. To make a fair comparison with related work, all architectures have been described and simulated in the developed tool.

First, we compare the proposed architectures with CBFP in terms of memory requirements and signal quality. In addition to the lower memory requirements, we will show how the co-optimized architecture produces a higher SQNR than CBFP. Second, we will compare the fabricated design with related work in terms of chip size and data throughput.

Table I shows a comparison of memory distribution between delay feedback units and intermediate buffers. 1-D architectures have fixed-point input, whereas 2-D architecture supports hybrid floating-point input. The table shows that the intermediate buffers used in CBFP consume a large amount of memory, which puts the co-optimized architecture in favor for 1-D processing. For 2-D processing, the co-optimized architecture also has lower memory requirements than hybrid floating point due to the buffer optimization. Figs. 7 and 8 present simulation results for the 1-D architectures in Table I. Fig. 7 is a simulation to compare SQNR when changing energy level in the input signal. In this case, the variations only affect $\mathrm{CBFP}_{\mathrm{low}}$ since scaling is applied later in the pipeline. Fig. 8 shows the result when applying signals with a large crest factor, i.e., the ratio between peak and mean value of the input. In this case, both CBFP implementations are strongly affected due to the large block size in the beginning of the pipeline. Signal statistics have minor impact on the hybrid floating-point architecture since every value is scaled individually. The SQNR for the co-optimized solution is located between hybrid floating point and CBFP since it uses a relatively small block size.

Table II shows an extended comparison between the proposed architectures and related work. The table includes two pipeline architectures using hybrid floating point, for 1-D signal processing (A) using a tailored datapath for exponent bits $E = 0, \ldots, 4$ and for 2-D signal processing (B) using a constant number of exponent bits $E = 4$. Then the proposed co-optimized architecture for 2-D signal processing (C), with a reduced hardware cost more comparable to the 1-D hybrid floating-point implementation. It uses block scaling in the initial butterfly unit and then hybrid floating point in the internal butterfly to show the low hardware cost for extending architecture (A) with support for 2-D processing.

The parallel architecture proposed by Lin *et al.* [10] uses BFP with a block size of 64 elements. The large block size affects the signal quality, but with slightly lower memory requirements compared to pipeline architectures. A pipeline architecture proposed by Bidet *et al.* [7] uses CBFP with a multipath delay commutator. The memory requirements are high due to the intermediate storage of data in the pipeline, which significantly affects the chip area. However, CBFP generates a higher SQNR than traditional BFP. The pipeline architecture proposed by Wang *et al.* [11] does not support scaling and is not directly comparable in terms of precision since SQNR depends on the input signal. The wordlength increases gradually in the pipeline to minimize the quantization noise, but this increases the memory requirements or, more important, the wordlength in arithmetic components and, therefore, also the chip area.

The proposed architectures have low hardware requirements and produce high SQNR using dynamic data scaling. They can easily be adopted to 2-D signal processing, in contrast to architectures without data scaling or using CBFP. The pipeline implementation results in a high throughput by continuous data streaming, which is shown as peak 1-D transforms in Table II.

VI. VLSI IMPLEMENTATION

A 2048-complex point pipeline FFT core using hybrid floating point and based on the Radix-2^2 decimation-in-frequency algorithm [5] has been designed, fabricated, and verified. This section presents internal building blocks and measurements on the fabricated ASIC prototype.

The butterfly units calculate the sum and the difference between the input sequence and the output sequence from the delay feedback. Output from the butterfly connects to the complex multiplier, and data is finally normalized and sent to the next FFT stage. The implementation of the delay feedbacks is a main consideration. For shorter delay sequences, serially connected flip-flops are used as delay elements. As the number of delay elements increases, this approach is no longer area and power efficient. One solution is to use SRAM and to continuously supply the computational units with data, one READ and one WRITE operation has to be performed in every clock cycle. A dual-port memory approach allow simultaneous READ and WRITE operations, but is larger and consumes more energy per memory access than single-port memories. Instead, two single-port memories, alternating between READ and WRITE each clock cycle could be used. This approach can be further simplified by using one single-port memory with double wordlength to hold two consecutive values in a single location, alternating between reading two values in one cycle and writing two values in the next cycle. The latter approach has been used for delay feedback exceeding the length of eight values. An area comparison can be found in [1].

A 2048-point FFT chip based on architecture (a) has been fabricated in a 0.35-μm, 5-ML CMOS process from AMI Semiconductor, see Fig. 9. The size of the core is 2632×2881 $\mu\mathrm{m}^2$ connected to 58

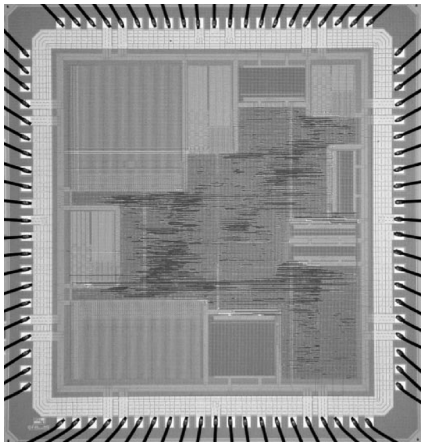

Fig. 9. Chip photo of the 2048 complex point FFT core fabricated in a 0.35-μm, 5-ML CMOS process. The core size is 2632×2881 μm^2.

input/output (I/O) pads and 26 power pads. The implementation requires 11 delay feedback buffers, one for each butterfly unit. Seven on-chip RAMs are used as delay buffers (approximately 49 Kb), while the four smallest buffers are implemented using flip-flops. Twiddle factors are stored in three ROMs containing approximately 47 Kb. The memories can be seen along the sides of the chip. The number of equivalent gates (two-input NAND) is 45900 for combinatorial area and 78300 for noncombinatorial area (including memories). The power consumption of the core was measured to 526 mW when running at 50 MHz and using a supply voltage of 2.7 V. The pipeline architecture produce one output value each clock cycle, or 37-K transforms per second running at maximum clock frequency. The 2-D FFT architecture (b) has been implemented on FPGA in [12].

VII. Conclusion

New dynamic data scaling architectures for pipeline FFTs have been proposed for both 1-D and 2-D applications. Based on hybrid floating point, a high-precision pipeline with low memory and arithmetic requirements has been constructed. A co-optimization between hybrid floating point and BFP has been proposed, reducing the memory requirement further by adding small intermediate buffers. A 2048 complex point pipeline FFT core has been implemented and fabricated in a 0.35-μm, 5-ML CMOS process, based on the presented scaling architecture and a throughput of 1 complex point/cc. The bidirectional pipeline FFT core, intended for image reconstruction in digital holography, has also been integrated on a custom designed FPGA platform to create a complete hardware accelerator for digital holographic imaging.

References

[1] T. Lenart and V. Öwall, "A 2048 complex point FFT processor using a novel data scaling approach," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2003, pp. 45–48.

[2] M. Gustafsson *et al.*, "High resolution digital transmission microscopy—a Fourier holography approach," *Opt. Lasers Eng.*, vol. 41, no. 3, pp. 553–563, Mar. 2004.

[3] *Digital Video Broadcasting (DVB); Framing Structure, Channel Coding, and Modulation for Digital Terrestrial Television*, ETSI EN 300 744 v1.4.1, Euro. Telecommun. Standards Inst., Sophia-Antipolis, France, 2001.

[4] J. W. Cooley and J. W. Tukey, "An algorithm for machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, Apr. 1965.

[5] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in *Proc. URSI Int. Symp. Signals, Syst. Electron.*, 1998, pp. 257–262.

[6] K. Kalliojärvi and J. Astola, "Roundoff errors is block-floating point systems," *IEEE Trans. Signal Process.*, vol. 44, no. 4, pp. 783–790, Apr. 1996.

[7] E. Bidet, C. Joanblanq, and P. Senn, "A fast single-chip implementation of 8192 complex point FFT," *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 300–305, Mar. 1995.

[8] F. Kristensen, P. Nilsson, and A. Olsson, "Reduced transceiver-delay for OFDM systems," in *Proc. Vehicular Technol. Conf., VTC* , 2004, pp. 1242–1245.

[9] C. Del Toso *et al.*, "0.5- μm CMOS circuits for demodulation and decoding of an OFDM-based digital TV signal conforming to the European DVB-T standard," *IEEE J. Solid-State Circuits*, vol. 33, no. 11, pp. 1781–1792, Nov. 1998.

[10] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, "A dynamic scaling FFT processor for DVB-T applications," *IEEE J. Solid-State Circuits*, vol. 39, no. 11, pp. 2005–2013, Nov. 2004.

[11] C.-C. Wang, J.-M. Huang, and H.-C. Cheng, "A 2K/8K mode small-area FFT processor for OFDM demodulation of DVB-T receivers," *IEEE Trans. Consumer Electron.*, vol. 51, no. 1, pp. 28–32, Feb. 2005.

[12] T. Lenart *et al.*, "Accelerating signal processing algorithm in digital holography using an FPGA platform," in *Proc. Int. Conf. FPT*, 2003, pp. 387–390.