Design of an 8192-point Sequential I/O FFT Chip

Yun-Nan Chang, *Member, IEEE*

Abstract—**This paper presents an efficient VLSI design of 8k-point pipeline fast Fourier transform (FFT) processor capable of producing the sequential order output. The proposed FFT architecture is derived based on the modified delay feed-forward data commutator, and processes the internal dual data streams in the real and imaginary alternate approach. Compared with the general pipeline FFT designs, ours can achieve full butterfly hardware efficiency such that the required number of adders can be reduced by a half. In order to generate the sequential output order sequence, this paper also proposes an efficient reorder buffer design which can be used to replace the last stage's commutator module for the saving of internal buffer. The FFT decomposition method for the proposed design is based on a new radix-$2^2 \times 2^2$ FFT algorithm such that only two and a half general complex-number multipliers are used. Finally, by proper data partition and allocation, the data storage required for data commutators and the output reorder buffer can both be efficiently realized by multi-bank single-port memory modules. The proposed FFT processor has been implemented and fabricated by 0.18μm CMOS process technology. The core size is about 8.74mm^2. This chip is suitable for digital video broadcasting (DVB) applications not only because it can perform the sequential input/output order 8k FFT, but it also consumes low power. To satisfy the DVB throughput requirement, this chip can run under the clock rate of 8MHz and the supply voltage of 1.33V, and only dissipates 20.6 mW.**

Index Terms—**FFT, DVB.**

I. Introduction

In recent years, due to the widespread use of the orthogonal frequency division multiplexing (OFDM) communication systems, how to design an efficient dedicated FFT circuit especially for the emerging OFDM applications is a very important issue. Many FFT designs have been proposed in the last decade [1]-[14], which can be categorized into non-pipeline and pipeline architectures. The non-pipeline FFT designs [1]-[3] utilize some centralized multiplier and butterfly arithmetic units to iteratively perform one stage of the computation after the other, and the intermediate results are stored in the central memory unit. This approach can achieve compact design but since the level of parallelism of this approach is restricted such that it is not suitable for high-throughput applications. On the other hand, the pipeline FFT processors [4]-[14] allocate dedicated data-path for each FFT stage such that it can achieve better throughput, and particularly suitable for the processing of continuous streaming input data.

The pipeline FFT designs can be further divided into two different classes. The first class is based on the so-called multi-path delay feed-forward (MDF) data commutator [4]-[8] which suffers both low hardware utilization and high storage space unless parallel input data streams are available. The other class of FFT designs is called single-delay feedback (SDF) FFT [9]-[14] which requires less internal buffer by circulating partial computed results back to the original stage to reuse the buffer space. They can also fully utilize the multiplier units although the utilization of their adder units that can be achieved is at most 50%. In addition to the data commutator style, the FFT algorithm which most of the early FFT designs adopt is based on the conventional radix-2 FFT algorithm. In order to reduce the silicon cost for the realization of the twiddle factor multiplication in FFT algorithm, several modified FFT algorithms including radix-4 [6],[9], radix-8, radix-2^2[10], and radix-2^3 [11] are proposed. These modified algorithms explore those trivial factors in the FFT algorithm such that their multiplications can be realized by dedicated constant multipliers instead of the general ones. In [15], a decimated dual-path delay fed-forward data commutator unit has been proposed. Combined with the new re-order buffer design, the resulted FFT architecture cannot only achieve fully butterfly unit utilization, but also support the normal input-output data order.

This paper presents an efficient implementation of a 8k-point FFT processor based on [15]. In addition, a new radix-$2^2 \times 2^2$ FFT algorithm is also proposed to reduce the number of general multipliers required. The remainder of this paper is organized as follows. Section II first presents the overall architecture of the proposed FFT design. In Section III, the detailed implementation of each FFT sub-module is addressed. Section IV presents the implementation and comparison results. Finally, some conclusion is given in Section V.

II. The proposed 8k-point FFT architecture

One of the main drawbacks of pipeline FFT designs compared with the non-pipeline is the low hardware utilization of the butterfly units. For the 8k-point pipeline FFT design, it will require 13 butterfly stages based on any radix-2^n decomposition. Therefore, it will require 52 real-number adders which represent a big portion of the entire FFT circuit. In order to reduce the adder cost, a new MDF design shown in Fig. 1 has been proposed [15]. Here the incoming data (d_j) are first divided into real (d_j^r) and imaginary (d_j^i) parts. Then they pass through two data commutator (DC) stages to form dual data streams which consist of even-index data followed by odd-index data. The real and imaginary parts of data are present in the stream in the alternate order. The remaining part of the FFT circuit is similar to the MDC FFT design with the exception that the data commutator and the arithmetic units will operate the real and imaginary data alternately. The function of data commutator, as shown in Fig. 2 is used to permute the data sequence in order to be fit for different stages of butterfly operation.

In Fig. 1, two twiddle factor multipliers are inserted between every two radix-2 stages; however, the actual implementation of these multipliers may be able to be simplified depending on what factors they involve. Furthermore, what

Manuscript received July 23, 2012; revised August 15, 2012.

Y.-N. Chang is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, 80424, Taiwan.

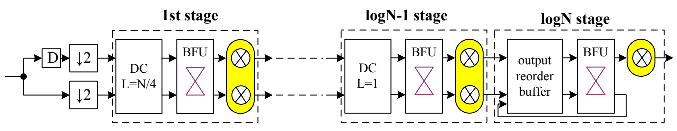

Fig. 1. The block diagram of the proposed pipeline FFT design which generates the normal order output sequence.

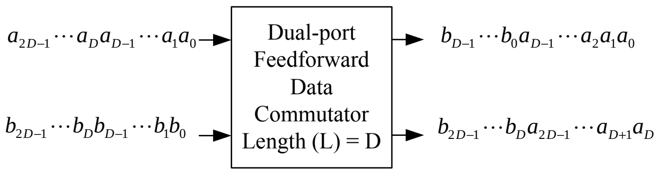
$a_{2D-1}\cdots a_D a_{D-1}\cdots a_1 a_0$
$b_{2D-1}\cdots b_D b_{D-1}\cdots b_1 b_0$
Dual-port Feedforward Data Commutator Length (L) = D
$b_{D-1}\cdots b_0 a_{D-1}\cdots a_2 a_1 a_0$
$b_{2D-1}\cdots b_D a_{2D-1}\cdots a_{D+1} a_D$

Fig. 2. The function of the data commutator (DC) with a depth of L.

twiddle factors are involved for each stage depends on the decomposition order applied to the DFT computation. For example, if the traditional decimation-in-frequency radix-2 FFT decomposition is applied for the 8k-point FFT, almost every stage requires general factor multipliers. The only exception will be the last several radix-2 stages which involve the multiplication of only few simple factors which can be realized by some dedicated constant multipliers. In order to reduce the multiplication strength, the trivial factors should be gathered to the same stages such that they can be realized by some dedicated constant multipliers. In the past, the idea of so-called radix-2^n FFT has been widely adopted in the past. In this paper, an extended method called radix-$2^2 \times 2^2$ FFT is proposed for the implementation of 8k-point FFT. Fig. 3 shows the data flow diagram of the simplified DFT after one stage of decomposition assuming N equals $N_1 \times N_2$.

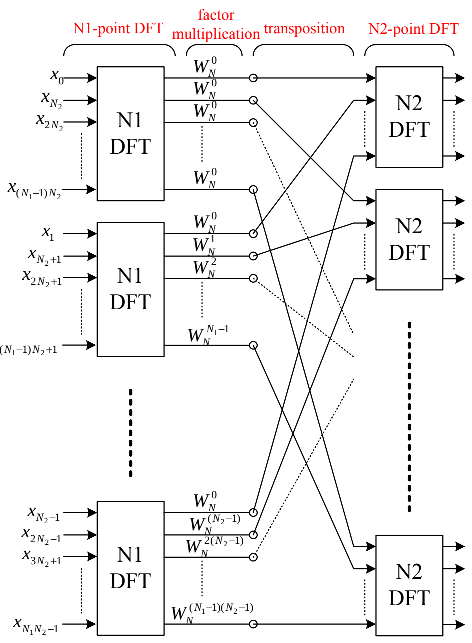

Fig. 3. The flow diagram of N-point DFT transformation after one-stage of decomposition asssuming N equals $N_1 \times N_2$.

A 8k pipeline FFT architecture can be divided into 13 stages. The detailed decomposition order for 8k-point FFT based on the proposed radix-$2^2 \times 2^2$ FFT approach is shown in Table I where each decomposition step is represented by three parameter N, N_1 and N_2 according to Fig. 3. The first level of decomposition is to recursively decompose the original DFT algorithm based on the function of 16-point DFT. After the first level, the function of each 16-point DFT will be further decomposed based on the 4-point DFT. Finally each of 4-point DFT computation is decomposed into two 2-point DFT operations which is equal to the basic butterfly function. Fig. 4 shows the block diagram of the resulted 8k-point pipeline FFT architecture. There are four types of multipliers being used in Fig. 4. The blocks labeled m-I and m-II denote two different general complex-number multipliers. The multiplier type m-IV involves the factors 1, -1, j and $-j$ such that it can be efficiently realized by simple multiplexer circuits. The multiplier type m-III involves the factors W_{16}^n, and can be realized by dedicated constant multipliers built by about 16 adders. The proposed radix-$2^2 \times 2^2$ FFT is superior to the radix-2^4 FFT [12] because the number of the constant multipliers can be reduced. For example, for 8k-point DFT, the radix-2^4 FFT requires six dedicated W_{16}^2 multipliers while the radix-$2^2 \times 2^2$ FFT only needs three.

TABLE I
PROPOSED DECOMPOSITION ORDER FOR 8K-POINT FFT.

Level	Level-1			Level-2			Level-3		
Step	N	N_1	N_2	N	N_1	N_2	N	N_1	N_2
Step 1	8k	2	4k	16	4	4	4	2	2
Step 2	4k	16	512						
Step 3	512	16	16						

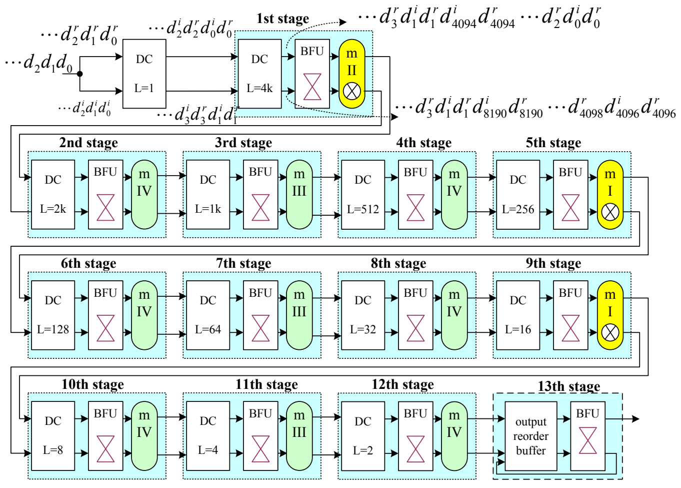

Fig. 4. The overall block diagram of the proposed 8k-point FFT architecture.

In the first level decomposition of 8k-point DFT, since the value of 8k is not the power of 16, we have to decompose it to the 4k-point DFT first by choosing either N_1 or N_2 to 2. Here we prefer the former choice due to the following reasons. First, the location of the general multipliers (stage 1, 5, and 9 for 8k-point FFT) will be closer to the input side compared with the other approach (stage 4, 8 and 12). Therefore, the cost of multipliers can be reduced since the word-length requirement for the stages closer to the input side will be generally shorter. The other reason is that for the resulted FFT flow chart, half of the first stage of butterfly unit outputs are multiplied by trivial factor W_{16}^0 which is equal to one. Based on this feature, the first stage multiplier can be further simplified as discussed in the following subsections.

III. Detailed FFT implementation

In the following, the implementation of each module shown in Fig. 4 is discussed in details.

A. *The arithmetic units*

There are three general complex-number multipliers required for 8k-point FFT architecture shown in Fig. 4. The second and third multipliers are used to multiply two butterfly operation results with associated twiddle factors in two cycles. Therefore, these type m-I multipliers blocks can be realized by one single complex-number multiplier to provide one factor multiplication each cycle. The details circuit is shown in Fig. 5(a) where the complex-number multiplier is implemented by three real-number multipliers with some extra adders based on

$$\begin{aligned} & (W_r + jW_i) \times (i_r + ji_i) \\ = & (W_r i_r - W_i i_i) + j(W_r i_i + W_i i_r) \\ = & [(W_r - W_i) \times i_r + (i_r - i_i) \times W_i] \\ & + j[(W_r + W_i) \times i_i + (i_r - i_i) \times W_i] \end{aligned} . \quad (1)$$

The other type of multiplier m-II shown in Fig. 4 can be realized even more simplified since one of the two outputs of the first stage butterfly unit is multiplied by the unity. Therefore, the multiplication of the other output can be realized by two real-number multipliers in time-multiplexing way as shown in Fig. 5(b).

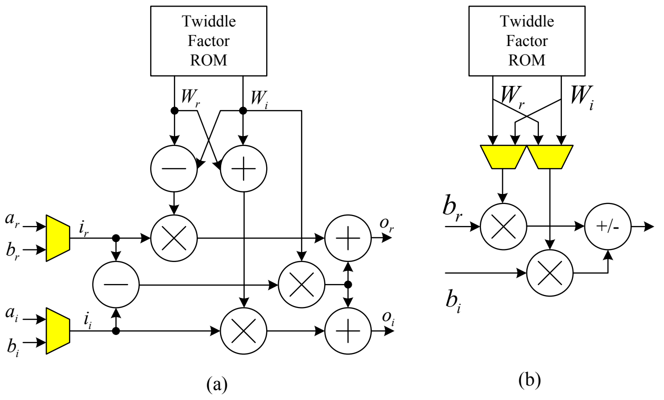

Fig. 5. The detailed circuit diagram of the multiplication unit used in the proposed FFT design: (a) type m-I multiplier, and (b) type m-II multiplier.

The butterfly unit is the fundamental building block of the pipeline FFT processors. In general, a basic radix-2 butterfly unit consists of one complex-number adder and one complex-number subtractor. However, for our pipeline FFT architecture shown in Fig. 4, the butterfly units just need to finish one butterfly operation for every two cycles. Therefore, except for the last stage, the butterfly unit architecture will be quite different from the common one. Fig. 6 shows the circuit diagram of two types of butterfly units used in our design. The first type of butterfly unit consisting of one real-number adder and subtractor, is used for the ordinary stages which do not need general multipliers. The real part of operation takes place alternatively with the imaginary part in the same hardware. On the other hand, the second type of butterfly unit is suitable for those stages equipped with multipliers. Since the multiplier is set to multiply one butterfly operation result with the factor each cycle, therefore, the entire complex-number result has to be generated at the same time. The use of different types of butterfly units can help reducing some flip-flops and multiplexors.

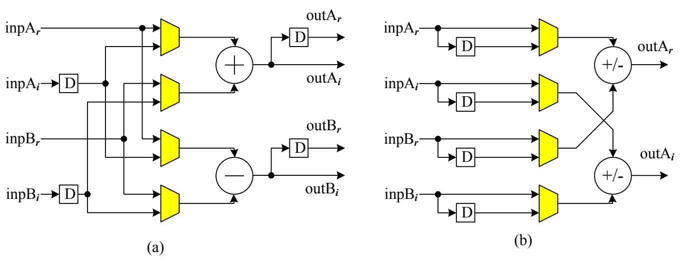

Fig. 6. The detailed circuit diagram of (a) type BFU-I and (b) type BFU-II butterfly unit.

B. *The data commutator design*

The function of the data commutator shown in Fig. 2 can be simply realized by some shift registers. However, this approach can only be suitable for the last several stages of commutators which do not buffer lots of data. For the first several stages, however, our design will implement the data commutator based on the use of two memory blocks of size $2L$ with some multiplexor control circuits. For the first L input data pairs, the switch is set to allow the data to pass through. Therefore, those inputs from the upper path are stored in memory block 0, and those from the other path are stored in block 1. The previous stored data in both memory blocks will be fetched, and sent to the output. For the next L input data pairs, the switch will be altered to allow the data flow change. Therefore, the inputs from the upper path are transmitted to the bottom output. The inputs from the bottom path are stored into the block 0. and those previous stored data in this block will be sent out. This same operation pattern will repeat again for next input sequence with length of 2L. Since for each memory block, at most one read and write operations are required for every two cycles, the memory block can be efficiently realized by the single-port on-chip SRAM.

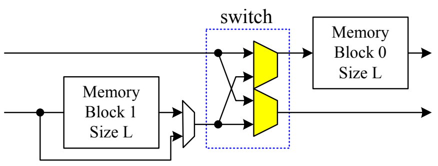

Fig. 7. The block diagram of data commutator with depth L based on two single-port memory banks.

C. *Design of twiddle factor ROM*

In addition to the design of the butterfly unit, factor multiplier, data commutator, and the reorder buffer, the twiddle factor ROM is another major part of the pipeline FFT processors. By exploring the symmetric properties of trigonometric functions, the original ROM which contains 8k sampling cosine and sine values between the angle $0°$ and $360°$ can now be simplified to contain only the sampling values between 0° and 45°. This reduction ratio of ROM is up to $\frac{1}{8}$ with only slight overhead of three adders and several multiplexors [16].

D. Design of the last radix-2 stage

The output order of transformed data produced by the pipeline FFT designs follows a special order called bit-reversed order. To reverse the output sequence back to the sequential order, the extra output reorder buffer that can hold all the entire 8192-point outputs has to be included. In order to reduce the buffer overhead, this paper proposes a new reordering approach by relocating the reorder buffer back to the input of the last radix-2 unit as shown in Fig. 4 such that the function of the reorder buffer can be integrated with the DC of the last stage. The detailed operation of the reorder buffer is described as follows. First, it will first store the entire set of 8k-point data generated by 12^{th} stage. Next, when it continues receiving the next 8k-point data set at the sample rate of two complex-number data every two cycles, simultaneously, it also fetches the previous set of data at the rate of two data per cycle and sends them to the butterfly unit. The data should be fetched in such an order that the butterfly unit can produce the sequential order of outputs. For the first 4096 cycles, the last-stage butterfly unit will generate a pair of outputs per cycle. One of the outputs belongs to the last half 4096-point results. Therefore, it has to be stored back to the reorder buffer in order to be output later. In summary, there will be at most eight buffer accesses including two input data write, four buffer data read (to provide the data operands for the butterfly unit), and two output data write-back operations for every two cycles.

TABLE II
THE DATA STORED AT THE ADDRESS j FOR EACH MEMORY BANK.

Bank number	Data set	
	even symbol	odd symbol
A0	$e_{4\times j+0}$	$o_{br(2\times j)}$
A1	$e_{4\times j+1}$	$o_{br(2\times j)+1}$
B0	$e_{4\times j+2}$	$o_{br(2\times j+1)}$
B1	$e_{4\times j+3}$	$o_{br(2\times j+1)+1}$
C0	$e_{4\times j+\frac{N}{2}}$	$o_{br(2\times j+\frac{N}{4})}$
C1	$e_{4\times j+1+\frac{N}{2}}$	$o_{br(2\times j+\frac{N}{4})+1}$
D0	$e_{4\times j+2+\frac{N}{2}}$	$o_{br(2\times j+1+\frac{N}{4})}$
D1	$e_{4\times j+3+\frac{N}{2}}$	$o_{br(2\times j+1+\frac{N}{4})+1}$

Therefore, direct implementation of this last stage may require double buffers, and each buffer requires multi-port on-chip SRAM with size of 8k words. To prevent the use of double buffer with large port count, multi-bank memory organization plus a minimum-conflict data distribution scheme is a very popular circuit design approach used to optimize the buffer implementation. Since the number of memory operations taking place in this buffer for each cycle is four, theoretically at least four banks of single-port memory modules will be required. This paper proposes one of a high efficient reorder buffer circuit based on eight banks of single-port memory. The size of each memory bank is 1K-word, and each bank will store the input data according to the allocation scheme shown in Table II. For the even symbol and odd symbol data set, the input samples are represented by e_j and o_j respectively. They have to be distinguished because they are written into the memory by different strategies. The even data sets are written sequentially while the odd data sets are written in an order similar to bit-reverse. In Table II, the function $br(j)$ is defined as the binary value of the bit-reverse representation of the 13-bit j. Since the even symbol and odd symbol enter the system alternately, a single buffer can be used and the input data will not overwrite the previous stored data that have yet to be processed.

The data placement scheme shown in Table II can not only avoid the use of double buffer, but also help distributing the memory operations equally into different banks. The memory read-write operations of the buffer can be illustrated in Table III. Here the symbol y_j represents the j^{th} output of the transformed sequence. It can be found that the all the read and write operations of each cycles take place in different banks such that the single port of memory is sufficient. Fig. 8 shows the overall architecture for the last radix-2 stage.

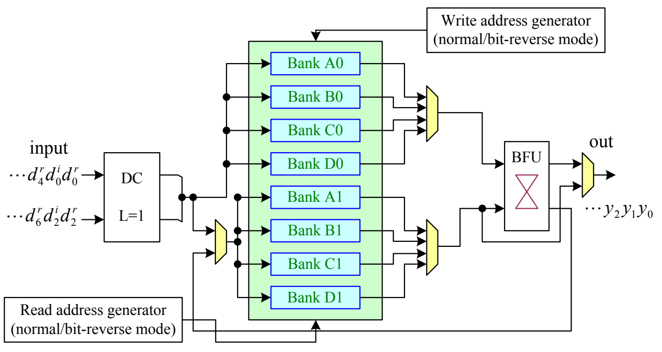

Fig. 8. The block diagram of the last stage of radix-2 unit.

IV. Experimental Results

Based on the proposed design methodology presented in the previous section, a pipeline 8k FFT processor has been implemented using 0.18μm CMOS technology. Fig. 9 shows the microphotography of the complete FFT chip. The gate count of this design excluding the memory blocks is about 124k and the core size of the layout is equal to 7.48mm^2. Taking the finite word-length effect into account, our design can achieve the signal-to-quantization-noise ratio (SQNR) over 96dB. The synthesis result shows the speed of our initial design can run up to 80MHz which is sufficient for the requirement of our target DVB applications. Therefore, no aggressive timing optimization techniques are further applied. As shown in this layout, since the required arithmetic units have been greatly reduced based the proposed methodology, the memory blocks has now become the major part of the entire chip. Especially, the reorder buffer of the last radix-2 stage occupied more than 35% of the silicon area not only because it has stored the entire 8k-point of data but its required word-length compared with other stages is also the largest. Fig. 10 shows the shmoo plot of this chip. This chip can run up to 33MHz. To meet the DVB specification, our chip can run at 8MHz under the supply voltage of 1.33V, and only dissipate 20.6 mW.

Table IV compares the proposed fabrication results with other published designs. The gate count number may vary a lot because many designs do not take their use of on-chip SRAM blocks into account. The frequency shown in

TABLE III
ILLUSTRATION OF THE MEMORY OPERATIONS IN DIFFERENT MEMORY BANKS.

	IN: odd symbol, OUT: even symbol									
Cycle	0	1	2	3	$\cdots$	2047	2048	2049	$\cdots$	8191
RD1	e_0	e_{4096}	e_{2048}	e_{6144}	$\cdots$	e_{8188}	e_2	e_{4098}	$\cdots$	y_{8190}
bank	A0	C0	A0	C0	$\cdots$	C0	B0	D0	$\cdots$	B1
RD2	e_1	e_{4097}	e_{2049}	e_{6145}	$\cdots$	e_{8189}	e_3	e_{4099}	$\cdots$	
bank	A1	C1	A1	C1	$\cdots$	C1	B1	D1	$\cdots$	
WT1		o_0	o_2	o_4	$\cdots$	o_{8188}	o_{8190}	o_1	$\cdots$	o_{8189}
bank		A0	C0	A0	$\cdots$	A0	C0	B0	$\cdots$	B0
WT2		y_{4096}	y_{4097}	y_{4098}	$\cdots$	y_{6142}	y_{6143}	y_{6144}	$\cdots$	
bank		A1	C1	A1	$\cdots$	A1	C1	B1	$\cdots$	
out		y_0	y_1	y_2	$\cdots$	y_{2046}	y_{2047}	y_{2048}	$\cdots$	y_{8190}
	IN: even symbol, OUT: odd symbol									
Cycle	0	1	2	3	$\cdots$	2047	2048	2049	$\cdots$	8191
RD1	o_0	o_{4096}	o_{2048}	o_{6144}	$\cdots$	o_{8188}	o_2	o_{4098}	$\cdots$	y_{8190}
bank	A0	B0	A0	B0	$\cdots$	B0	C0	D0	$\cdots$	C1
RD2	o_1	o_{4097}	o_{2049}	o_{6145}	$\cdots$	o_{8189}	o_3	o_{4099}	$\cdots$	
bank	A1	B1	A1	B1	$\cdots$	B1	C1	D1	$\cdots$	
WT1		e_0	e_2	e_4	$\cdots$	e_{8189}	e_{8190}	e_1	$\cdots$	e_{8189}
bank		A0	B0	A0	$\cdots$	A0	B0	C0	$\cdots$	C0
WT2		y_{4096}	y_{4097}	y_{4098}	$\cdots$	y_{6142}	y_{6143}	y_{6144}	$\cdots$	
bank		A1	B1	A1	$\cdots$	A1	B1	C1	$\cdots$	
out		y_0	y_1	y_2	$\cdots$	y_{2046}	y_{2047}	y_{2048}	$\cdots$	y_{8190}

TABLE IV
COMPARISON RESULTS OF DIFFERENT 8K-POINT FFT DESIGNS. (*: THE REDUCED CORE SIZE BY OMITTING THE REORDER BUFFER.)

	Proposed	[2]	[5]	[9]	[11]	[8]	[13]	[14]
Gate count	124k	NA	1.5M	139k	1.3M	700k	988k	600k
Technology	0.18	0.18	0.5	0.35	0.6	NA	NA	NA
FFT method	radix-$2^2 \times 2^2$	radix-8	radix-4	radix-4	split-radix	radix-4	radix-2/4/8	NA
Architecture	MDC	non-pipeline	MDC	SDF	SDF	MDC	SDF	pipeline
Area (mm^2)	8.74 (4.5*)	4.84	100	33.75	107	NA	NA	NA
Normalized area	8.74	4.84	11.57	8.92	12.54	NA	NA	NA
Power(mW)	20.6	25.2	600	535	650	NA	NA	NA
Frequency for DVB	8MHz	20MHz	20MHz	16MHz	20MHz	NA	NA	NA
Output order	natural	either	reverse	reverse	reverse	reverse	reverse	reverse
Additional buffer (word)	0	8k $\times 2$	8k	8k	8k	8k	8k	reverse

the table represents the operating clock rate suited for DVB applications. As shown in Table IV, our proposed design is the smallest with the exception of some non-pipeline FFT designs. It should be noted that all pipeline FFT designs in the literature generate the transformation sequence with the bit-reverse order. To obtain the normal sequential output-order sequence, they will require an additional buffer with size of 8k words. As for those non-pipeline designs, since they store all the transformation results on the memory block, the output order can be set for any form. However, for continuous streaming data input, they will require two additional buffers with 8k words as mentioned before. The issue of additional buffer requirement is often neglected in the past. Since these buffers are used to store the final transformation outputs which usually have larger word-length compared with the input data, the actual silicon size of these buffers can be very huge. Therefore, by considering the extra overhead, the cost of the proposed design can outperform all the past FFT designs. The reorder buffer of the last radix-2 stage occupied more than 35% of the silicon area, and the core size of our chip can be reduced to $4.5mm^2$ by excluding the last-stage reorder buffer.

V. CONCLUSION

This paper proposed an efficient 8k-point pipeline FFT design which can generate the sequential order output sequence. The design uses only about half number of adders compared with the other pipeline circuits. In order to provide a transformed output sequence with normal order, a novel output reorder buffer mechanism is also proposed which can be efficiently integrated with the function of the data commutator and realized with a single buffer which consists

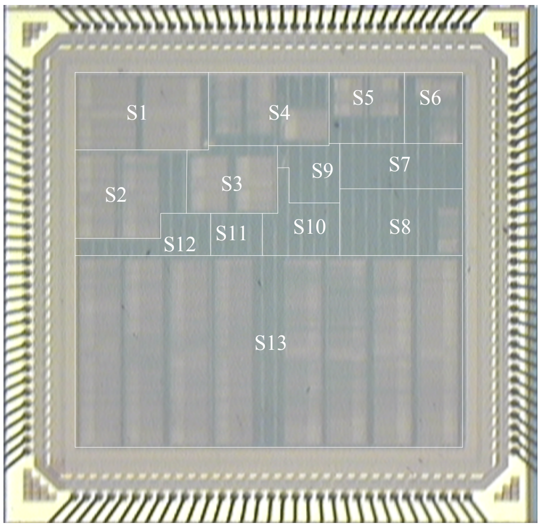

Fig. 9. The microphotograph of the complete FFT chip.

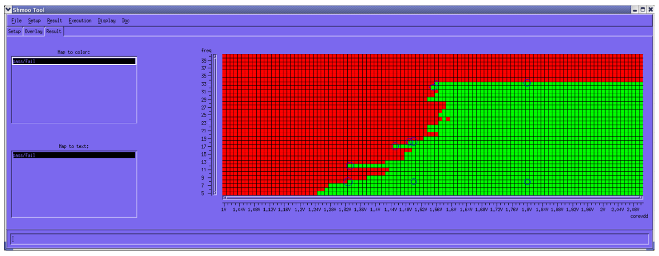

Fig. 10. The shmoo plot of the proposed fabricated FFT chip.

of eight single-port on-chip SRAM memory banks. By using the proposed radix-$2^2 \times 2^2$ FFT decomposition method, the 8k-point FFT processor will only require two and a half general complex-number multipliers. The final fabrication results show that our 8k-point FFT architecture is not only power-efficient, but also represents the most compact design.

According to our fabrication result, the reorder buffer occupies the most portion of the entire chip. Since at most two memory read and two write operations are required per cycle, in future, the reduction of the bank numbers of the reorder buffer from eight to four will be pursued.

ACKNOWLEDGMENT

The authors would like to thank National Chip Implementation Center (CIC) in Taiwan for chip fabrication.

REFERENCES

[1] B. M. Baas, “A low power, high performance, 1024-point FFT processor,” *IEEE Journal of Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, Mar 1999.
[2] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, “A dynamic scaling FFT processor for DVB-T applications,” *IEEE Journal of Solid-State Circuits*, vol. 39, no. 11, pp. 2005–2013, Nov 2004.
[3] C.-L. Wey, W.-C. Tang, and S.-Y. Lin, “Efficient VLSI implementation of memory-based FFT processors for DVB-T application,” in *IEEE Computer Society Annual Symposium on VLSI*, Porto Alegre, Brazil, 2007, pp. 98–106.
[4] G. Bi and E. Jones, “A pipelined FFT processor for word-sequential data,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 1982–1985, Dec. 1989.
[5] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn, “A fast single-chip implementation of 8192 complex point FFT,” *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 300–305, Mar 1995.
[6] E. E. Swartzlander, W. K. W. Young, and S. J. Joseph, “A radix-4 delay commutator for fast Fourier transform processor implementation,” *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 5, pp. 702–709, Oct. 1984.
[7] M. Ayinala, M. Brown, and K. K. Parhi, “Pipelined parallel FFT architectures via folding transformation,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1068–1081, 2012.
[8] S. H. Park, D. H. Kim, D. S. Han, K. S. Lee, S. J. Park, and J. R. Choi, “Sequential design of a 8192 complex point FFT in OFDM receiver,” in *Proc. IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, Seoul, Korea, Aug. 1999, pp. 262–265.
[9] 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. on Consumer Electronics*, vol. 51, no. 1, pp. 28–32, Feb 2005.
[10] S. He and M. Torkelson, “Designing pipeline FFT processor for OFDM (de)modulation,” in *URSI International Symposium on Signals, Systems, and Electronics*, Sep. 1998, pp. 257 – 262.
[11] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, “A new VLSI-oriented FFT algorithm and implementation,” in *Proc. Eleven Annual IEEE International Conference on Advanced System Integrated Circuits*, Rochester, New York, Sep. 1998, pp. 337–341.
[12] J.-Y. Oh and M.-S. Lim, “Fast Fourier transform processor based on low-power and area-efficient algorithm,” in *Proc. IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, Fukuoka, Japan, Aug. 2004, pp. 198 – 201.
[13] T.-H. Tsai and C.-C. Peng, “Design and implementation of a FFT/IFFT soft IP generator for OFDM system,” in *International Conference on Consumer Electronics*, Las Vegas, NV, Jan. 2005, pp. 385–386.
[14] S. Y. Park, N. I. Cho, S. U. Lee, K. Kim, and J. Oh, “Design of 2k/4k/8k-point FFT processor based on CORDIC algorithm in OFDM receiver,” in *IEEE Pacific Rim Conference on Communications, Computers and signal processing*, vol. 2, Victoria, Canada, Aug. 2001, pp. 457–460.
[15] Y.-N. Chang, “An efficient VLSI architecture for normal I/O order pipeline FFT design,” *IEEE Transactions on Circuits and Systems-II*, vol. 55, no. 12, pp. 1234–1238, Dec 2008.
[16] S.-Y. Lee, C.-C. Chen, C.-C. Lee, and C.-J. Cheng, “A low-power VLSI architecture for a shared-memory FFT processor with a mixed-radix algorithm and a simple memory control scheme,” in *Proc. of 2006 IEEE ISCAS*, Island of Kos, Greece, May 2006, pp. 157–160.