Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

An efficient radix-2 fast Fourier transform processor with ganged butterfly engines on field programmable gate arrays

Zhen-guo MA[†], Feng YU[†‡], Rui-feng GE, Ze-ke WANG

(*Department of Instrument Engineering, Zhejiang University, Hangzhou 310027, China*)

[†]E-mail: mzgzju@yahoo.com.cn; osfengyu@zju.edu.cn

Received July 21, 2010; Revision accepted Jan. 6, 2011; Crosschecked Feb. 28, 2011

Abstract: We present a novel method to implement the radix-2 fast Fourier transform (FFT) algorithm on field programmable gate arrays (FPGA). The FFT architecture exploits parallelism by having more pipelined units in the stages, and more parallel units within a stage. It has the noticeable advantages of high speed and more efficient resource utilization by employing four ganged butterfly engines (GBEs), and can be well matched to the placement of the resources on the FPGA. We adopt the decimation-in-frequency (DIF) radix-2 FFT algorithm and implement the FFT processor on a state-of-the-art FPGA. Experimental results show that the processor can compute 1024-point complex radix-2 FFT in about 11 μs with a clock frequency of 200 MHz.

Key words: Ganged butterfly engine (GBE), Radix-2, Fast Fourier transform (FFT), Field programmable gate array (FPGA)
doi:10.1631/jzus.C1000258 **Document code:** A **CLC number:** TN91

1 Introduction

Fast Fourier transform (FFT) (Cooley and Tukey, 1965), as a fast version of discrete Fourier transform (DFT), has benefited science and engineering communities in numerous ways. As an efficient algorithm for computing DFT, FFT plays an important role in many signal processing domains, such as television broadcasting (Camarda *et al*., 2009), frequency domain modulation (Chalermsuk *et al*., 2008), and spectral analysis (Thoen *et al*., 2009).

Nowadays, FFT processors are based mainly on the application specific integrated circuit (ASIC) (Pitkanen and Takala, 2009), digital signal processing (DSP) (Sun and Yu, 2009), or field programmable gate arrays (FPGA) (Kee *et al*., 2009). Compared with ASIC and DSP, FPGA has the advantages of more flexibility, less design time, and lower cost. As FPGA advances, the amount of parallelism to be exploited grows rapidly with the increasing resources on the FPGA. The performance of FPGA is growing fast, and FPGA-based designs outperform CPU-based designs in some applications (Asano *et al*., 2009). Much of the parallelism of the FFT algorithm can be exploited, and it is a good way to implement efficient FFT processors on the FPGA.

FFT architectures can be classified into two main categories: memory-based architecture and pipeline architecture. Several pipeline architectures have been proposed (He and Torkelson, 1996; Garrido *et al*., 2009; Liu *et al*., 2011), such as radix-2 multi-path delay commutation, radix-2 single-path delay feedback, and radix-2^2 single-path delay feedback. The pipeline architectures employ many processing engines (PEs) to achieve higher performance; in contrast, the memory-based architecture usually requires only one butterfly PE, as well as some block memories for storing input and intermediate data. Yeh and Truong (2007) analyzed the area and speed performances of some memory-based FFT processors that feature low speed with low hardware consumption. The main approaches to making FFT processors perform much better are more resource consumption, a higher-radix algorithm, and more parallelism. The resources are limited on a certain FPGA, and

‡ Corresponding author

higher-radix algorithms increase complexity and reduce flexibility. Thus, we improve the performance of the FFT processor by exploiting more parallelism with better resource utilization on the FPGA. In this paper, we present an efficient FFT processor, named the GBE processor, as it employs four ganged butterfly engines. The GBE architecture balances the high speed and low resource usage requirements, and can be well matched to the placement of the resources on FPGA.

2 Fast Fourier transform algorithm

FFT is an efficient algorithm for computing the DFT of sizes that are positive integers power of 2. The DFT function of $X(k)$, which is an N-point sequence of $x(n)$, is defined as

$$X(k)=\sum_{n=0}^{N-1}x(n)\,\mathrm{e}^{-\mathrm{j}(2\pi/N)kn},\quad 0\le k\le N-1.$$

FFT can be used to effectively decompose and compute a DFT by using the symmetry and periodicity of the complex sequence. The algorithm for radix-2 decimation-in-frequency (DIF) FFT is defined below, where $W_N=\mathrm{e}^{-\mathrm{j}2\pi/N}$ is often referred to as the twiddle factor:

$$X(k)=\sum_{n=0}^{N/2-1}\left(x(n)+(-1)^k x\left(n+\frac{N}{2}\right)\right)W_N^{kn},\quad 0\le k\le N-1.$$

Fig. 1 shows the schematic of basic radix-2 butterfly computation, which contains two additions and one multiplication. The FFT algorithm can be calculated by replicating radix-2 butterfly computations.

In N-point DIF FFT computation, the input data of $x(n)$ is in natural order, while the output data is in bit reversed order. The N-point FFT computation is done in $\log_2 N$ stages, and each stage contains $N/2$ butterfly computations. The total amount of butterfly computation is $(N/2)\log_2 N$. Fig. 2 gives an example of the 16-point radix-2 DIF FFT algorithm.

The radix-2 algorithm yields the smallest butterfly unit, which allows for greater flexibility in the design space. High radix algorithms reduce the number of operations, but increase complexity and reduce flexibility. For example, a typical radix-4 FFT algorithm does not support the FFT algorithms whose points are not positive integers of 4, and radix-4 PE is more complex than radix-2 PE. We adopt the DIF radix-2 FFT algorithm in our design.

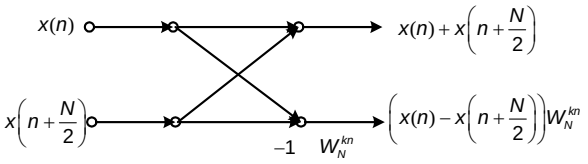

Fig. 1 Radix-2 butterfly computation

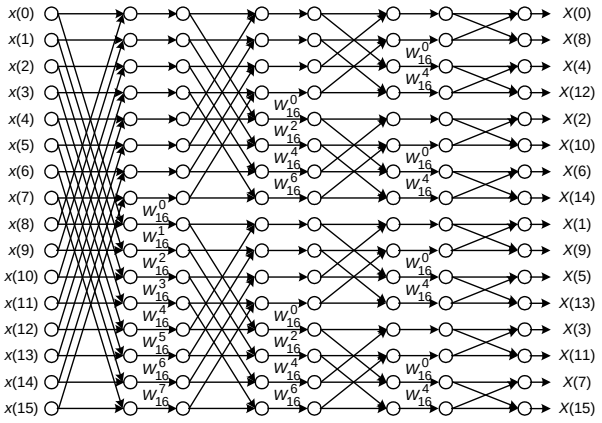

Fig. 2 An example of the 16-point radix-2 decimation-in-frequency (DIF) FFT algorithm

3 Architectures of fast Fourier transform processors

3.1 Memory-based architecture

A typical radix-2 memory-based architecture contains one radix-2 butterfly PE and one or more RAMs (Fig. 3). Compared with RX2-B1 (Fig. 3a) that processes only one data sample, RX2-B2 (Fig. 3b) can process two data samples in parallel. RX2-B2 achieves twice the speed performance compared with RX2-B1, while the resources of the ROM and PE are the same, as is the total size of the RAMs. The difference between these two architectures is that RX2-B2 has two separate RAMs, while RX2-B1 has only one, whose size is twice that of RX2-B2. For N-point RX2-B1 or RX2-B2 FFT architecture, the total size of RAMs is N, and the ROM size is $N/2$.

3.2 Pipeline architecture

A typical N-point radix-2 pipeline FFT architecture has $\log_2 N$ butterfly PEs (Fig. 4). The N-point FFT computation is done in $\log_2 N$ stages, and in each

stage RAMs and ROMs are used to store the intermediate data and the twiddle factors. In this way, the pipeline FFT processor can simultaneously do butterfly computations, load input data, and unload the result.

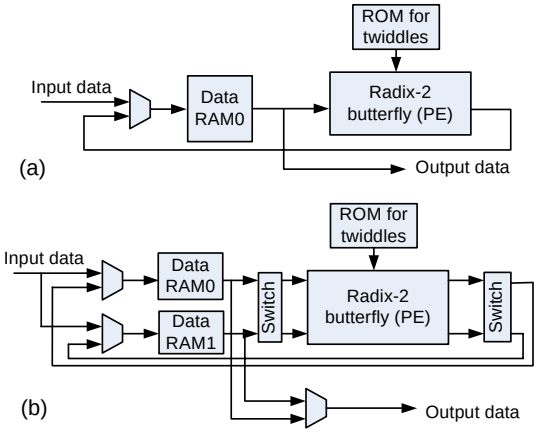

Fig. 3 Typical radix-2 memory-based architecture
(a) RX2-B1; (b) RX2-B2

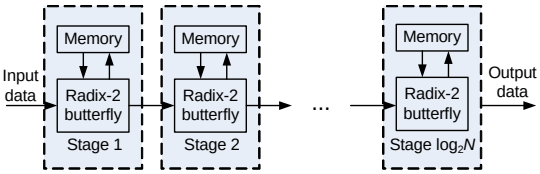

Fig. 4 Typical radix-2 pipeline architecture

In modern FPGAs, RAM and ROM can be made up of either the embedded block memory or the slices. It consumes a lot of slices to form large memories. In a typical pipeline FFT architecture, the sizes of RAM and ROM decrease by half as the stage number increases by one. Unfortunately, the size of block memory in FPGA is fixed. For example, if the block memory size is fixed to 1024-point, four separate 256-point storage elements consume four block memories, although their total size is 1024-point. Implementing small RAMs using block memories in pipeline architectures has proven wasteful (He and Torkelson, 1998). In an N-point pipeline FFT processor, the resources of the RAM, ROM, and butterfly PE increase when the value of $\log_2 N$ increases. The pipeline large point FFT is not efficient on the FPGA, as it consumes many resources including block memories and slices.

Typical memory-based FFT processors consume fewer resources, but the speed is quit low. Pipeline architectures achieve higher speed at the cost of more resources. There is always a balance of speed and resource in FFT processor design.

3.3 The proposed architecture

For more efficient resource utilization on FPGA, we should exploit the parallelism more. The parallelism of FFT computation can be exploited in two ways: pipelined units (parallelism in multiple stages) and parallel units (parallelism within a stage). To this end, we propose an efficient radix-2 GBE processor (Fig. 5).

A GBE processor contains four butterfly PEs, eight data RAMs, and two twiddle factor ROMs. After all the data are written into RAMs 1–4, multiplexer0 selects the data from switch 2. Each switch contains four multiplexers (MUX), as shown in Fig. 6.

For N-point FFT computation, the size of each RAM is $N/4$. Fig. 7 shows the data flow graph of 16-point FFT. The four data in each RAM are from low address to high address (00 01 10 11). The size of

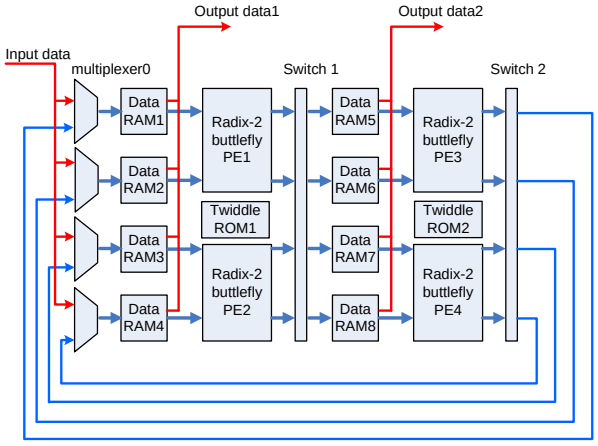

Fig. 5 Architecture of the ganged butterfly engine (GBE) processor

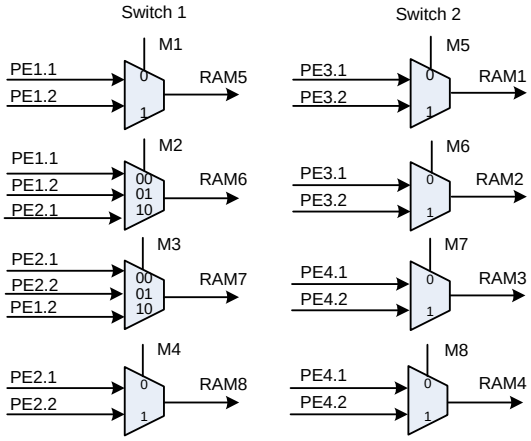

Fig. 6 Architecture of the two switches in Fig. 5

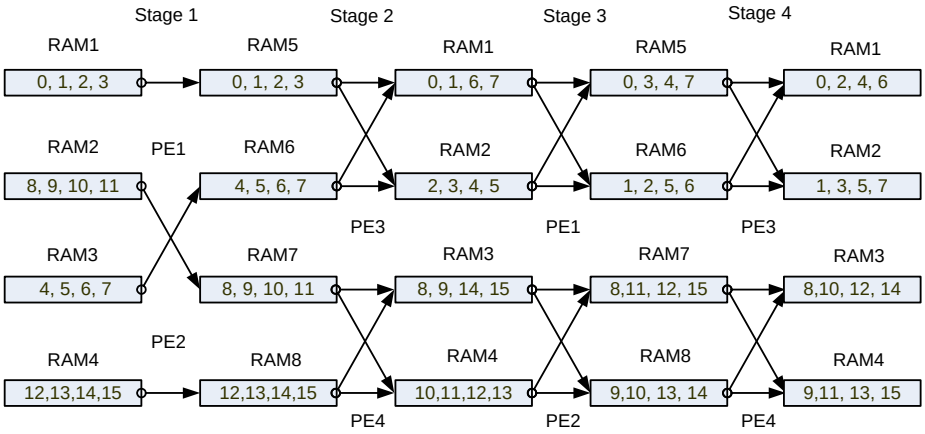

Fig. 7 Data flow graph of 16-point FFT

the data in RAMs is also stored from low to high addresses. After the end stage of stage 4, the result data are stored in RAMs 1–4 in bit reversed order. Additional RAMs are required for reordering to arrange the result data in a more natural way.

In the GBE processor, each stage takes $N/4$ clock cycles. For N=16, each stage contains four clock cycles, where two-bit counters (w_1, w_0) and (r_1, r_0) are used to count the clock cycles at each stage. Interestingly, the control signals for MUXs, RAMs, and ROMs can be represented in terms of the counters at each stage (Table 1).

Table 1 Control signals of MUX, RAM, and ROM with N=16

S	M1	M2	M3	M4	RAM WA		ROM1 RA
					5, 7	6, 8	
1	0	10	10	1	w_1w_0	w_1w_0	$0w_1w_0$; $1w_1w_0$
3	w_0	$0\ \overline{w}_0$	$0w_0$	$\overline{w}_0$	w_1w_0	$w_1\ \overline{w}_0$	$w_0$00
S	M5	M6	M7	M8	RAM WA		ROM2 RA
					1, 3	2, 4	
2	r_1	$\overline{r}_1$	r_1	$\overline{r}_1$	r_1r_0	$\overline{r}_1\ r_0$	$r_1r_0$0
4	0	1	0	1	r_1r_0	r_1r_0	000

S: stage. WA: write address; RA: read address

Twiddle factors are pre-computed and stored in ROMs. The N-point FFT needs $N/2$ twiddle factors. In the first stage, the data in ROM1 with address $0w_1w_0$ are read to PE 1, and the data with address $1w_1w_0$ are read to PE 2. PEs 1 and 2 share the same twiddle factor after the first stage, as well as PEs 3 and 4. The write addresses of the RAMs are shown in Table 1, and the read addresses are always from low to high.

Table 2 shows the control signals of 64-point FFT. Only the first stage is different from the other stages, and one can easily extend this to any pattern N. Thus, the control circuitry is simply composed of the counters. For structural simplicity and regularity, we use simple finite state machines (FSMs) to produce the control signals and increase flexibility. The GBE processor is scalable in point size and supports smaller point FFT computations.

Table 2 Control signals of MUX, RAM, and ROM with N=64

S	M1	M2	M3	M4	RAM WA		ROM1 RA
					5, 7	6, 8	
1	0	10	10	1	w_3w_2 w_1w_0	w_3w_2 w_1w_0	$0w_3w_2w_1w_0$; $1w_3w_2w_1w_0$
3	w_2	$0\ \overline{w}_2$	$0w_2$	$\overline{w}_2$	w_3w_2 w_1w_0	$w_3\ \overline{w}_2$ w_1w_0	$w_2w_1w_0$00
5	w_0	$0\ \overline{w}_0$	$0w_0$	$\overline{w}_0$	w_3w_2 w_1w_0	w_3w_2 $w_1\ \overline{w}_0$	$w_0$0000
S	M5	M6	M7	M8	RAM WA		ROM2 RA
					1, 3	2, 4	
2	r_3	$\overline{r}_3$	r_3	$\overline{r}_3$	r_3r_2 r_1r_0	$\overline{r}_3\ r_2$ r_1r_0	$r_3r_2r_1r_0$0
4	r_1	$\overline{r}_1$	r_1	$\overline{r}_1$	r_3r_2 r_1r_0	r_3r_2 $\overline{r}_1\ r_0$	$r_1r_0$000
6	0	1	0	1	r_3r_2 r_1r_0	r_3r_2 r_1r_0	00000

S: stage. WA: write address; RA: read address

Normal butterfly computation is as shown in Fig. 1. In contrast, in the GBE processor the two input data of the butterfly PEs are sometimes reversed. We guarantee the correctness of the result using the

negative of the twiddle factors (Fig. 8). The control signal can also be represented in terms of the counters. For example, in 64-point FFT computation (Table 2), the reverse order of input data occurs when w_3=1 in stage 3, r_2=1 in stage 4, w_1=1 in stage 5, and r_0=1 in stage 6.

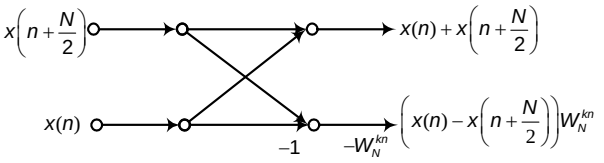

Fig. 8 Reverse input of radix-2 butterfly computation

Fig. 9 shows the parallelism of the GBE processor. When N/8 data have been written into RAMs 1–4, PEs 1 and 2 start to do butterfly computation. After butterfly computation, the intermediate result data are written into RAMs 5–8 through switch 1. When N/8 data have been written into RAMs 5–8, the data are soon read out to PEs 3 and 4. When N/8 data have been written into RAMs 1–4, the data are read out to PEs 1 and 2. The process repeats until it reaches the end stage of FFT. After the end stage, the result data are unloaded. We name the PEs as ganged butterfly engines, as they do two-stage butterfly computations simultaneously and the processing mechanism is analogous to a gear. Thanks to the method of employing four GBEs, the parallelism in FFT computation is well exploited.

Compared with typical memory-based FFT architectures, the GBE architecture has the noticeable advantage of high speed by employing four PEs, and exploits more parallelism between the PEs.

Another important advantage is that the GBE processor can be well matched to the placement of the resources on the FPGAs (Fig. 5). On the FPGAs, the slices and multiplication macros are placed between the block memories. Hence, the GBE processor can be readily mapped and routed at a high frequency on the FPGA.

4 Results and analysis

We used Xilinx ISE 10.3 and MATLAB R2006a. We configured the FPGA and used ChipScope software (in ISE 10.3) to capture the relational signals. Then we compared the result data of the GBE processor with the result data computed using MATLAB.

Speed performance was measured in clock cycle, from loading the first data to storing the last result data (Fig. 9). The latency of the GBE processor is computed with Eq. (1):

$$T_{\text{total}} \approx \left(\frac{N}{8}+T_{\text{pe\_delay}}\right)\log_2 N+\frac{N}{4}+\frac{3N}{8}, \qquad (1)$$

where $T_{\text{pe\_delay}}$ is the delay of PE.

The result data in the RAMs are in reversed order and it takes 3N/8 clock cycles to make the reordering for natural order. The speed performances of two typical memory-based FFT processors are defined below, excluding the cycles for reordering:

$$T_{\text{RX2-B1}} \approx (N+T_{\text{pe\_delay}})\log_2 N+N, \qquad (2)$$

$$T_{\text{RX2-B2}} \approx (N/2+T_{\text{pe\_delay}})\log_2 N+N/2. \qquad (3)$$

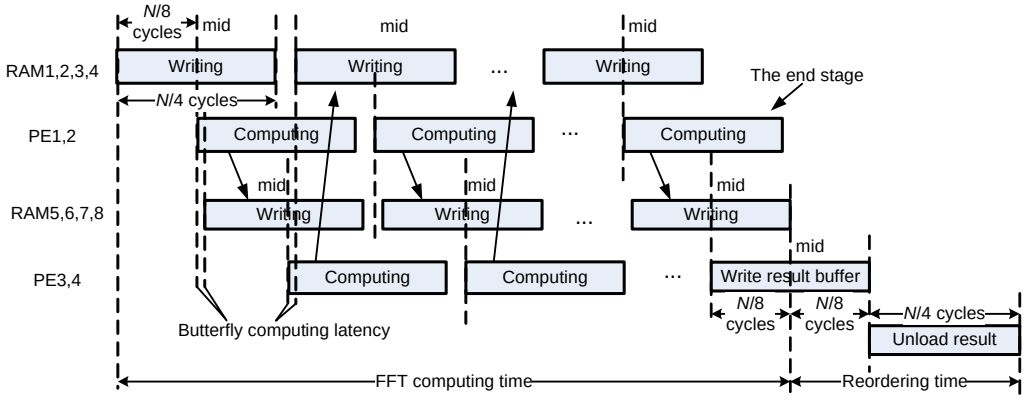

Fig. 9 Diagram of parallelism of the ganged butterfly engines (GBE) processor

Table 3 shows the speed performance and block memory resources of the FFT processors without reordering. The latency is computed with a PE clock delay of 0. The data and the twiddle factor are the 16-bit real and imaginary parts of a complex value, respectively, and the block memory for each is configured as a dual-port 1024×36 (36K) memory bank, which can store 1024-point 32-bit data. When the required memory size is larger than 1024-point, more block memories are needed.

Table 3 Speed and memory resources comparison

Point size	FFT design	Block memory*	Latency** (cycles)
1024	RX2-B1	2 (1/1)	11 264
	RX2-B2	3 (2/1)	5632
	GBE	10 (8/2)	1536
2048	RX2-B1	3 (2/1)	24 576
	RX2-B2	3 (2/1)	12 288
	GBE	10 (8/2)	3328
4096	RX2-B1	6 (4/2)	53 248
	RX2-B2	6 (4/2)	26 624
	GBE	12 (8/4)	7168
8192	RX2-B1	12 (8/4)	114 688
	RX2-B2	12 (8/4)	57 344
	GBE	24 (16/8)	15 360

* (RAM/ROM); ** With a PE clock delay of 0

Table 3 shows that the number of block memories increases with the increase of point size N, when the required memory size is larger than 1024-point. When $N\geq4096$, the total size of block memories of the GBE architecture is 3N-point, which is twice that of RX2-B1/RX2-B2. The latency of GBE is about one fourth that of RX2-B2.

Yeh and Truong (2007) introduced a metric to measure the performance between two processors: speed performance gain versus area gain (SP/A). The ratio is measured between two designs and determined using Eq. (4):

$$\mathrm{SP/A}=\frac{\text{difference in clock cycle}}{\text{IHU}}, \tag{4}$$

where IHU stands for increase in hardware utilization and is measured in percent, and the difference is also measured in percent.

A decrease in the number of clock cycles between the two designs is considered positive. In the computation of IHU, the slice, multiplier, and block memory resource account for one-third, respectively. We fine-tune the metric by replacing IHU with RU (resource utilization) using Eq. (5):

$$\mathrm{SP/R}=\frac{\text{difference in clock cycle}}{\text{RU}}. \tag{5}$$

RU is composed of the utilization of three parts: control unit (CU), PE, and block memory (BM). Each part accounts for one-third utilization. Thus, RU= (CU_diff+PE_diff+BM_diff)/3, where CU_diff, PE_ diff, and BM_diff are differences in CU, PE, and BM, respectively. An increase of consumed resources between two designs is considered positive. The differences and RU are measured in percent.

RX2-B2 consumes the same size of block memory and the same number of PEs as RX2-B1, while the processing latency of RX2-B2 is half that of RX2-B1. Hence, the SP/R between RX2-B2 and RX2-B1 is about 2. CU is simple and regular in the GBE processor, as the control signal is represented in terms of a counter at each stage. The increasing utilization of the CU between GBE and RX2-B2 is slightly larger than 2, and we assume it to be 3. The GBE processor has four PEs, while the RX2-B2 processor has only one. The memory size of RX2-B2 is 1.5N-point, and the memory size of GBE is 3N-point. The SP/R between GBE and RX2-B2 is

$$\mathrm{SP/R}=\frac{N/2\cdot\log_2 N+N/2}{N/8\cdot\log_2 N+N/4}\Big/\left(\frac{3}{3}+\frac{4}{3}+\frac{2}{3}\right), \tag{6}$$

which is about 1.238 when N=4096. In Eq. (6), the 3/3, 4/3, and 2/3 are the differences of CU, PE, and BM, respectively.

Radhouane *et al.* (2000) presented a continuous flow FFT implementation for VLSI architecture to minimize the memory requirement. It takes $N/2\cdot\log_2 N$ clock cycles to process and $N/2$ clock cycles to store the result data with the memory size of 2N-point. The SP/R between GBE architecture and Radhouane's is

$$\mathrm{SP/R}=\frac{N/2\cdot\log_2 N+N/2}{N/8\cdot\log_2 N+N/4}\Big/\left(\frac{3}{3}+\frac{4}{3}+\frac{1.5}{3}\right), \tag{7}$$

which is about 1.31 when N=4096. In Eq. (7), the 3/3, 4/3, and 1.5/3 are the differences of CU, PE, and BM, respectively.

It is shown that the GBE architecture is also attractive to large-point FFT VLSI design. The GBE architecture has the noticeable advantage of more efficient resource utilization, compared with typical memory-based FFT architectures.

The N-point typical pipeline FFT processor contains $\log_2 N$ PEs and many simpler control units. The memory resource utilization is not efficient for large-point pipeline FFT processors on the FPGA. The processing latency of a typical N-point FFT pipeline processor is N cycles, as it deals with the streaming data. In the GBE processor, unloading of the current frame and loading of the next frame can occur simultaneously. Therefore, the latency of the GBE processor can be computed using Eq. (8):

$$T_{\text{delay}} \approx (N/8 + T_{\text{pe\_delay}}) \log_2 N + N/4. \quad (8)$$

Thus, the latency between two frames in the 1024-point GBE processor is about 1536 cycles with a PE clock delay of 0. The GBE architecture is more efficient in resource utilization, especially at a larger point size.

In our experiment, we adopted the floating-point radix-2 FFT algorithm. The GBE processor can compute 1024-point complex FFT with the process of reordering in about 11 μs with a clock frequency of 200 MHz.

5 Conclusions

We present a novel method to implement an efficient radix-2 FFT processor on the FPGA. We compared our GBE architecture with some other FFT architectures. It is shown that the GBE processor achieves high speed, high clock frequency, and more efficient resource utilization. The main contribution of our FFT architecture is exploiting more parallelism in radix-2 FFT processors by employing four ganged butterfly engines. Another significant advantage is that the GBE architecture can be well matched to the placement of the resources on modern FPGAs, which makes high-frequency implementation easier. The GBE architecture is very suitable and efficient for radix-2 FFT processor designs on FPGAs.

References

Asano, S., Maruyama, T., Yamaguchi, Y., 2009. Performance Comparison of FPGA, GPU and CPU in Image Processing. Int. Conf. on Field Programmable Logic and Applications, p.126-131. [doi:10.1109/FPL.2009.5272532]

Camarda, F., Prevotet, J.C., Nouvel, F., 2009. Implementation of a Reconfigurable Fast Fourier Transform Application to Digital Terrestrial Television Broadcasting. Int. Conf. on Field Programmable Logic and Applications, p.353-358. [doi:10.1109/FPL.2009.5272266]

Chalermsuk, K., Spaanenburg, R.H., Spaanenburg, L., Seutter, M., Stoorvogel, H., 2008. Flexible-Length Fast Fourier Transform for COFDM. 15th IEEE Int. Conf. on Electronics, Circuits and Systems, p.534-537. [doi:10.1109/ICECS.2008.4674908]

Cooley, J.W., Tukey, J.W., 1965. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, **19**:297-301. [doi:10.1090/S0025-5718-1965-0178586-1]

Garrido, M., Parhi, K.K., Grajal, J., 2009. A pipelined FFT architecture for real-valued signals. *IEEE Trans. Circ. Syst. I: Reg. Papers*, **56**(12):2634-2643. [doi:10.1109/TCSI.2009.2017125]

He, S.S., Torkelson, M., 1996. A New Approach to Pipeline FFT Processor. Proc. 10th Int. Parallel Process. Symp., p.766-770. [doi:10.1109/IPPS.1996.508145]

He, S.S., Torkelson, M., 1998. Design and Implementation of 1024-point Pipeline FFT Processor. Custom Integrated Circuits Conf., p.131-134. [doi:10.1109/CICC.1998.694922]

Kee, H., Bhattacharyya, S.S., Petersen, N., Kornerup, J., 2009. Resource-Efficient Acceleration of 2-Dimensional Fast Fourier Transform Computations on FPGAs. Third ACM/IEEE Int. Conf. on Distributed Smart Cameras, p.1-8. [doi:10.1109/ICDSC.2009.5289356]

Liu, X., Yu, F., Wang, Z.K., 2011. A pipelined architecture for normal I/O order FFT. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **12**(1):76-82. [doi:10.1631/jzus.C1000234]

Pitkanen, T., Takala, J., 2009. Low-Power Application-Specific Processor for FFT Computations. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, p.593-596. [doi:10.1109/ICASSP.2009.4959653]

Radhouane, R., Liu, P., Modlin, C., 2000. Minimizing the Memory Requirement for Continuous Flow FFT Implementation: Continuous Flow Mixed Mode FFT (CFMM-FFT). Proc. IEEE Int. Symp. on Circuits and Systems, **1**:116-119. [doi:10.1109/ISCAS.2000.857040]

Sun, T.Y., Yu, Y.H., 2009. Memory Usage Reduction Method for FFT Implementations on DSP Based Embedded System. IEEE 13th Int. Symp. on Consumer Electronics, p.812-815. [doi:10.1109/ISCE.2009.5156962]

Thoen, D.J., Bongers, W.A., Westerhof, E., Oosterbeek, J.W., de Baar, M.R., van den Berg, M.A., van Beveren, V., Burger, A., Goede, A., Graswinckel, M.F., *et al.*, 2009. Fast Fourier Transform Based Diagnostics for Spectral Characterization of Millimeter Waves in Tokamaks. 34th Int. Conf. on Infrared, Millimeter, and Terahertz Waves, p.1-2. [doi:10.1109/ICIMW.2009.5325586]

Yeh, H.G., Truong, G., 2007. Speed and Area Analysis of Memory Based FFT Processors in a FPGA. Wireless Telecommunications Symp., p.1-6. [doi:10.1109/WTS.2007.4563313]