IMPLEMENTATION OF APPLICATION-SPECIFIC DSP FOR OFDM SYSTEMS

Jeong H. Lee, Jong H. Moon, and Myung H. Sunwoo
School of Electrical and Computer Engineering, Ajou University
San 5, Wonchun-Dong,Youngtong-Ku, Suwon, 442-749, KOREA
Email: sunwoo@ajou.ac.kr

1. INTRODUCTION

Multicarrier modulation techniques, such as, OFDM (Orthogonal Frequency Division Modulation) and DMT (Discrete Multi-Tone), have received great attention in high-speed data communication systems [1]. In communication systems, OFDM has become popular in various standards, for example, local area network (WLAN), digital audio broadcasting (DAB), digital video broadcasting (DVB), very high-speed digital subscriber line (VDSL), and powerline communications (PLC) standards. In general, embedded communication systems require large computational power and low power consumption. Therefore, ASIC (Application-Specific Integrated Circuit) solutions have been widely used to implement communication systems. However, there have been strong demands to implement communication systems using programmable processors, because of their flexibility. Hence, the market of ASDSP compromising advantages of both ASIC and DSP is growing.

This paper describes the implementation of the proposed application-specific instructions and their DSP architecture, which can implement high-speed FFT/IFFT, scrambler/ descrambler, convolutional encoder interleaver/deinterleaver, and puncturing blocks without using a hardwired ASIC. The proposed architecture can execute the algorithm blocks of OFDM modem systems [2] faster than the existing DSP chips in terms of execution cycles.

This paper is organized as follows. Section 2 describes the proposed instructions and their hardware architecture for high-speed FFT. Section 3 describes the implementation and Section 4 explains performance comparisons. Finally, Section 5 contains concluding remarks.

2. NEW INSTRUCTION AND HARDWARE ARCHITECTURE FOR OFDM BLOCKS

This section describes instructions and their hardware architecture of ASDSP which are used to efficiently implement OFDM modem systems.

2.1. Two approaches for FFT acceleration

In this section, two approaches for FFT acceleration are described. First, FAGU (FFT Address Generation Unit) is proposed to reduce extra cycles during FFT computation. Second, application specific instructions and the modified DALU architecture are proposed.

2.1.1 Proposed FAGU to reduce FFT computation cycles

FFT computation consists of several stages according to the number of FFT points. Each stage consists of groups, and a group is decomposed into small butterflies. Hence, according to the FFT algorithms, two or three nested DO loops are required to execute the whole computation using existing DSP chips. Equation (1) represents the computation cycles of an N-point FFT. L represents the number of butterfly computation cycles, M represents the DO loop latency and α is referred to as the number of cycles taken to calculate butterfly addresses. In Equation (1), $\log_2 N$ of the first term represents the number of stages and N/2 represents the number of butterflies for each stage. Hence, only the first term of equation (1) is the FFT computation cycles, and the other terms are extra cycles to calculate butterfly addresses. However, in FFT computation, the order of input and output data

Acknowledgements: This work is supported in part by the NRL (National Research Laboratory) program of MOST, in part by HY-SDR research center under the ITRC program of MIC, in part by IDEC

ISBN: 89-86522-78-0 94460©KEES

addresses during the whole FFT computation is determined by the number of FFT points. Hence, the extra cycles can be reduced by automatically generating the addresses.

$$(L \times N/2) \log_2 N + M \times (N-1) + \log_2 N + \alpha. \qquad (1)$$

This paper proposes FAGU to automatically generate input data addresses of FFT butterflies. The proposed FAGU requires very small hardware consisting of four registers, two counters, three adders and three comparators. The counters and comparators are used to calculate group and loop counts, and the adders and registers are used to generate input data addresses of FFT butterflies.

2.1.2 Proposed instructions and DALU architecture

Fig. 1 represents how to compute a butterfly with the proposed DALU. Herein, a_r and b_r represent real data, w_r represents real twiddle factor, a_i and b_i represent imaginary data and w_i represents imaginary twiddle factor.

The new SBUTTERFLY (Subtraction-BUTTERFLY) instruction, which is one cycle instruction, is proposed to perform two multiplications and then two additions and one subtraction concurrently in cycle ①. In the same way, we propose the new ABUTTERFLY (Addition-BUTTERFLY) instruction to execute operations in cycle ②. The scheme using the SBUTTERFLY instruction and the ABUTTERFLY instruction performs one radix-2 butterfly in 2 cycles as shown in Fig. 1.

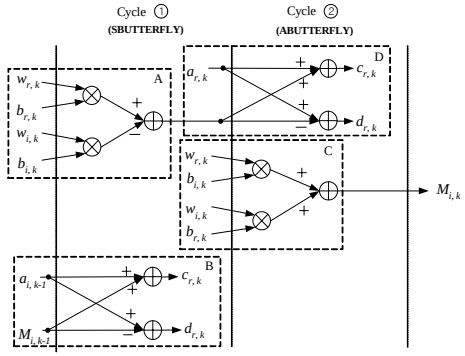

Fig. 1. Proposed flow graph of a radix-2 butterfly

Fig. 2 represents the proposed DALU architecture that can support SBUTTERFLY and ABUTTERFLY instructions. Carmel DSP uses the similar flow graph with Fig. 1. To execute operations in the box A, Carmel DSP performs two multiplications using Mul1 and Mul2 in Fig. 2. The result of a Mul1 cannot be loaded to Acc2 and vice versa. Hence, the result of Mul1 bypasses Acc1 to perform subtraction which is executed using Acc2. Hence, one of the accumulators in two MACs is not used, and two ALUs are required to execute an addition and a subtraction, which are the operations in the box B in cycle ①.

Herein, two MACs, one ALU and switching logic are used. Inserting the switching logic between Muls and Adders, two inputs can be directly loaded into the accumulators. Hence, operations in the box A in Fig. 1 can be executed using only Mul1, Mul2 and Adder1 while the operations in the box B are executed using Adder2 and Adder3.

2.2 Instructions for scrambling, convolutional encoding, and puncturing

Scrambling/descrambling, convolutional encoding, punctu -ring, and interleaving/deinterleaving mainly require bit manipulation. Basic operations of bit manipulation include repeated shifts, XOR, bit insertion, and bit extraction. To accelerate these operations, we propose SCB, CONV, and PUNC instructions. SCB and CONV can perform shift and bitwise XORs of several input data in a clock cycle. PUNC inserts selected bits from a source data register into a destination data register. For each bit set in the 8-bit mask, the bit in the source register is copied to the selected bit position of the destination operand by the 16-bit mask. The bits not selected are unaffected. A scramble sequence of various communication standards can be generated using the SCB and PUNC instructions. Likewise, convolutional encoding can be performed using the CONV and the PUNC

instructions. The proposed DSP shows better performance than existing DSPs. The additional hardware for these instructions is of only 1,700 gates.

2.3 Proposed ASDSP architecture

Fig 3. shows the implemented ASDSP architecture. The proposed architecture has one program memory, two data memories, PCU, DALU, AGU, and FAGU. Each of the internal word lengths is 16 bit. The instruction pipeline consists of six stages: pre-fetch, fetch, decode, execute1, execute2, and execute3.

FAGU is disabled to minimize power consumptions when ASDSP executes general instructions and it is enabled when ASDSP executes FFT instructions. The FFT flag register indicates whether FAGU is active or not. To control the FFT flag register, the FFT #N instruction is proposed. Herein, N means the FFT points. After the number of FFT points is specified, only ABUTTERFLY and SBUTTERFLY instructions are repeatedly executed as the number of butterflies of the whole FFT computation. These two instructions are stored in the instruction registers at the first execution, and then two instructions are fetched from the instruction register, not from the program memory during FFT computation. An assembly code for a 64-point FFT is as follows

```
FFT #N
ABUTTERFLY
SBUTTERFLY.
```

Commercial DSPs require a large number of instructions to perform FFT computations, such as ADD, SUB, LOAD, STORE and MAC, and also require several hundred lines of the assembly code. On the other hand, the proposed ASDSP requires only three instructions. Hence, power consumption can be dramatically reduced during FFT computation, since only three instruction fetches from program memory are required.

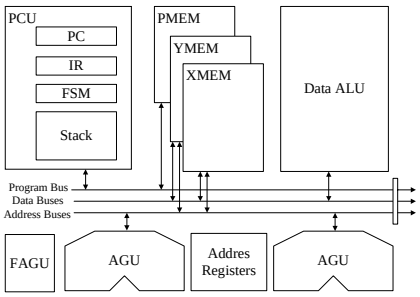

Fig. 2. Proposed DALU architecture

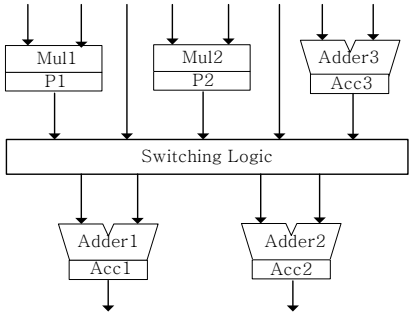

Fig. 3. Proposed ASDSP architecture

3. IMPLEMENTATIONS

The proposed architecture has been modeled by VHDL and logic synthesis has been performed using the SEC 0.18 ㎛ technology. The total gate count of the ASDSP is about 80,000 gates. Besides, FAGU consists of only 5,600 gates which occupy small area. Fig. 4 shows the layout of ASDSP. The maximum delay is about 3.54 ㎱, and thus, the maximum operating frequency is about 280 MHz. The proposed ASDSP has been thoroughly verified using the iPROVE FPGA board having Xilinx vertex II. The SQNR value of the FFT output is about 71 dB which shows very good performance for communication systems. The proposed ASDSP has 19 arithmetic instructions, 9 logical instructions, 5 move instructions, 7 program control instructions and 11 application specific instructions.

4. PERFORMANCE EVALUATION

Table I shows the performance comparisons among DSPs for FFT computations [3, 4]. Extra cycles for ASDSP are only 8, while other DSPs are relatively large. For N = 64, (64 / 2) x $\log_2 64$ butterflies must be computed. Since the proposed architecture takes 2 cycles per one butterfly, 2 x (64 / 2) x $\log_2 64$ + 8 = 390 clock cycles are needed for completing a 64-point FFT. The proposed DSP shows better performance than the other architectures as shown in Table I. Moreover, the proposed architecture has smaller area compared with the other DSPs. Carmel DSP and TMS320c62X having

more hardware resources require more cycles compared with the proposed DSP by 8% to 53%. Moreover, Carmel DSP and TMS320C62X are VLIW (Very Long Instruction Word) cores and have large hardware size. For example, Carmel DSP has 73 bit program bus and TMS320C62X has 32 bit program bus while the proposed ASDSP has 16 bit program bus. Hence, VLIW cores consume much larger power.

Conventional ASIC FFT processors can execute the FFT algorithm faster than the proposed ASDSP since most of ASIC FFT processors use radix-4 butterfly computation. However, the computation speed of the proposed ASDSP is sufficient to satisfy the various communication standards. Moreover, conventional ASIC FFT processors support only limited number of FFT points, while the ASDSP can support from the 64 point to 8192 point FFT computation [5].

Table I. Performance comparisons of FFT computations

DSPs	Required cycles
Carmel DSP	$(N+10)\log_2 N + 3N/2 - 29$
TMS320C62X	$(4N/2)\log_2 N + 7\log_2 N + N/4 + 9$
ASDSP	$(2N/2)\log_2 N + 8$

Table II. Comparisons of FFT computation times

Standards	Number of Point	Require computation time of standard (μs)	Computation time of ASDSP (μs)
WLAN	64	4	1.4
DAB	511	62	16.5
	2048	246	80.5
DVB-T	2048	231	80.5
	8192	924	321.5
VDSL	4096	250	175.5

Table II shows the required computation times of one OFDM symbol for various standards and computation times of the proposed ASDSP. The estimated time of our DSP for a 64-point FFT is 392 x 3.54 ns = 1.39 μs. Thus, it can satisfy the requirement of the WLAN standard (802.11a) in which the FFT computation should be completed within 4 μs. In the DVB-T system, an 8,192-point FFT should be completed in 924 μs. The estimated time of our DSP is 106,504 x 3.54 ns = 321.5 μs. Hence, the proposed architecture can satisfy the requirements of most OFDM standards, such as WLAN, DVB-T, etc.

5. CONCLUSIONS

This paper describes the implementation and verification of the proposed DSP architecture for OFDM modem systems. The implemented DALU has only dual MACs and one ALU and FAGU to efficiently provides input data addresses for FFT. The architecture has been modeled by VHDL and logic synthesis has been using the SEC 0.18μm standard cell library. The implemented DSP consists of 80,000 gates and FAGU has about 5,600 gates. The proposed ASDSP has been thoroughly verified using the iPROVE PPGA board and it can be used in various OFDM and DMT modem systems.

6. REFERENCES

[1] N. Weste and D. J. Skellern, "VLSI for OFDM," *IEEE Commun. Mag*., pp. 127-131., Oct. 1998.

[2] Byung G. Jo, Byung S. Son, Myung H. Sunwoo, and Yong Serk Kim, "A High-Speed FFT Processor For OFDM SYSTEMS," in *Proc. IEEE Int. Symp. on Circuits and Syst.*, 2002, pp. 281-284.

[3] *CARMEL DSP Core Data Sheet*, Infineon Technologies Inc., 1999.

[4] *TMS320C62xx User's Manual*, Texas Instruments Inc., Dalla, TX, 1997.

[5] Drey Enterprise Inc., *Jaguar II Variable-Point (8-1024) FFT/IFFT Specification*, 1998