LOW-POWER APPLICATION-SPECIFIC PROCESSOR FOR FFT COMPUTATIONS

Teemu Pitkänen and Jarmo Takala

Tampere University of Technology
Department of Computer Systems
P.O.Box 553, FIN-33101 Tampere, Finland

ABSTRACT

In this paper, we describe a processor architecture tailored for radix-4 and mixed-radix FFT algorithms, which have lower arithmetic complexity than radix-2 algorithms. The processor is based on transport triggered architecture and several optimizations have been used to improve the energy-efficiency. The processor has been synthesized on a 130nm standard cell technology and analysis show that a programmable solution can possess energy-efficiency comparable to a fixed-function ASIC.

Index Terms— Discrete Fourier transforms, Application specific integrated circuits, Digital signal processors, Parallel architectures

1. INTRODUCTION

Recently fast Fourier transform (FFT) has gained popularity as OFDM has been used in several wireless and wireline communication systems, e.g., IEEE 802.11a/g, 802.16, VDSL, and DVB. FFT implementations are often based on Cooley-Tukey radix-2 FFT algorithms due to the regularity and principal simplicity of the computations. However, arithmetic complexity can be reduced by exploiting radix-4 FFT algorithms but these algorithms can only support power-of-four sequence lengths. A solution supporting power-of-two lengths with lower complexity than radix-2 is mixed-radix FFT where radix-4 and radix-2 computations can be combined.

Traditionally FFT has been realized as fixed-function VLSI implementation since it provides better energy-efficiency and performance than software implementations. Such implementations have recently been reported, e.g., in [1, 2], implementations based on radix-2 FFT are reported and examples of radix-4 FFTs can be found, e.g., from [3, 4]. Implementations based on mixed-radix algorithms are reported in [5, 6].

Recently, software implementations have become preferable due to the flexibility but the energy-efficiency of programmable architectures has been poor compared to dedicated hardware structures. However, the energy-efficiency can be improved by customizing the architecture towards to the application domain.

In this paper, we describe a processor architecture tailored for FFT computations. Several optimizations have been used to improve the energy-efficiency of the processor. This paper combines results from our previous papers [7, 8] and shows that a programmable solution can possess energy-efficiency comparable to fixed-function ASIC. The processor is tailored for radix-4 and mixed-radix FFT algorithms and supports several transform lengths.

This work has been supported in part by the Academy of Finland under project 205743 and the Finnish Funding Agency for Technology and Innovation under research funding decision 40163/07.

2. FFT ALGORITHMS

In this paper, we have used the in-place decimation-in-time (DIT) radix-4 FFT algorithm with in-order-input, permuted output as given, e.g., in [9] and corresponding mixed-radix algorithm. The radix-4 algorithm can be formulated as follows:

$$F_{2^{2n}} = R_{4^n}\left[\prod_{s=n-1}^{0}\left(I_{4^s}\otimes F_4\otimes I_{4^{n-s-1}}\right)\left(T_{4^{s+1},4^n}\otimes I_{4^{n-s-1}}\right)\right]; \quad (1)$$

$$F_4 = \begin{pmatrix} 1 & 1 & 1 & 1\\ 1 & -j & -1 & j\\ 1 & -1 & 1 & -1\\ 1 & j & -1 & -j \end{pmatrix}; \quad (2)$$

$$R_{4^n} = \prod_{k=n-2}^{0} I_{4^k}\otimes P_{4^{(n-k)},4} \quad (3)$$

where $\otimes$ denotes tensor product, I_N is identity matrix of order N, and R_N is a permutation matrix based on stride-by-S permutation matrix of order N, $P_{N,S}$, defined as [10]

$$\left[P_{N,S}\right]_{mn} = \begin{cases} 1, & \text{iff } n = (mS \bmod N) + \lfloor mS/N \rfloor \\ 0, & \text{otherwise} \end{cases}, \quad m,n = 0,1,\ldots,N-1\,. \quad (4)$$

The matrix $T_{k,N}$ contains twiddle factors $W_N^m = e^{j2\pi m/N}$ as follows

$$T_{k,N} = \operatorname{diag}\Big(\omega(0)^0,\omega(0)^1,\omega(0)^2,\omega(0)^3, \omega(1)^0,\omega(1)^1,\omega(1)^2,\omega(1)^3,\ldots, \omega(k/4-1)^2,\omega(k/4-1)^3\Big); \quad (5)$$

$$\omega = R_{N/4}(W_N^0, W_N^1, \ldots, W_N^{(N/4-1)})^T \quad (6)$$

An example of signal flow graph of this algorithm is depicted in Fig. 1a). Power-of-two FFT can be supported by using mixed-radix algorithm consisting of radix-4 and radix-2 computations as follows

$$F_{2^{2n+1}} = S_{2^{(2n+1)}}\left(I_{4^n}\otimes F_2\right)U_{2^{(2n+1)}}\cdot \left[\prod_{s=n-1}^{0}\left(I_{4^s}\otimes F_4\otimes I_{2^{2n-2s-1}}\right)\left(T_{4^{s+1},4^n}\otimes I_{2^{2n-2s-1}}\right)\right]; \quad (7)$$

$$F_2 = \begin{pmatrix} 1 & 1\\ 1 & -1 \end{pmatrix}; \quad (8)$$

$$S_{2^{2n+1}} = \left(I_2\otimes R_{4^n}\right)P_{2^{2n+1},2} \quad (9)$$

where twiddle factors for the last radix-2 processing column are defined as

$$U_N = S_N^T\left(I_{N/2}\oplus\operatorname{diag}\left(W_N^0, W_N^1, \ldots, W_N^{N/2-1}\right)\right) \quad (10)$$

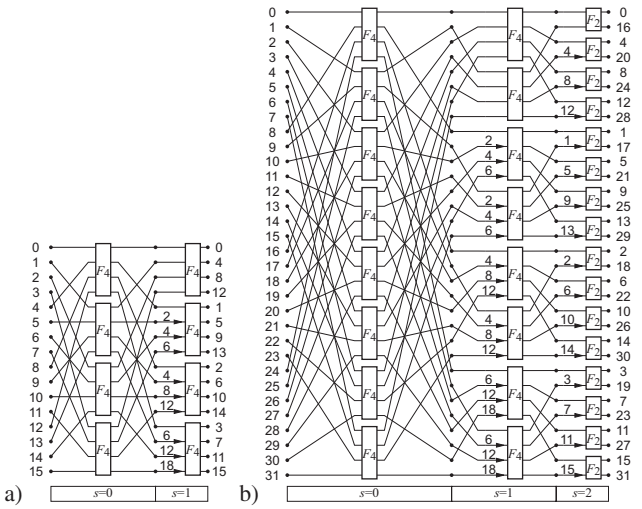

Fig. 1. Signal flow graph of a) 16-point radix-4 FFT and b) 32-point mixed-radix FFT. A constant k in the signal flow graph represents a twiddle factor W_{32}^{k}.

where $\oplus$ denotes matrix direct sum. An example of signal flow graph of the mixed-radix algorithm based on radix-4 and radix-2 computations is shown in Fig. 1b).

3. PROCESSOR ORGANIZATION

The proposed processor is based on transport triggered architecture (TTA) [11] , which a class of statically programmed instruction-level parallelism (ILP) architectures reminding very long instruction word (VLIW) architectures. In TTA programming model, the program specifies only the data transfers in the interconnection network and actual operations occur as "side-effect" of data transports. Operands to a function unit are input through ports and one of the ports acts as a trigger. Whenever data is moved to the trigger port, function unit initiates operation execution. When the input ports are registered, the operands for the operation can be stored into the registers in earlier instruction cycles and a transport to the trigger port starts the operation with the operands stored into the registers. Thus the operands can be shared between different operations of a function unit, which reduces the data traffic in the interconnection and the need for temporary storage in register file or data memory.

A TTA processor consists of a set of function units and register files containing general-purpose registers. These structures are connected to an interconnection network, which connects the input and output ports of the resources. The architecture can be tailored by adding or removing resources. Moreover, special function units with user-defined functionality can be easily included. Such units are the key to energy-efficient implementations; customization of the function units according to the specific characteristics of the application is effective means to reduce power consumption without reducing the performance. Such units also reduce the instruction overhead, thus they reduce the power consumption due to instruction fetch. In this work, we have used several custom-designed units tailored for FFT application.

A processor customized for FFT should support complex data type and here we simply spit the native word into two parts representing real and imaginary part of complex number. We also introduced

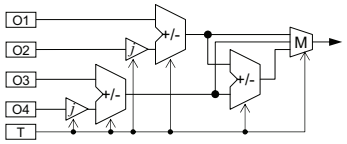

Fig. 2. Block diagram of complex adder unit.

complex-valued multiplication and addition units. The complex-valued adder unit computes any of the four-operand addition defined in 4-point DFT, F_4, in (2) and the structure is shown in Fig. 2. In this unit, the operation code is used as the trigger port, thus once the four operands (summands) are transferred to input ports, four results can be triggered by transferring the opcode to trigger code without the need to transfer the operands.

In order to save memory storage, in-place computations are used, i.e., results from butterfly computations are stored to memory locations where the operands were read. This implies that for N-point FFT only N-memory locations are used. FFT computations contain also special access patterns and computing the addresses for memory accesses with traditional instruction set would take several instructions. Significant savings in power consumption can be obtained by using special address generation units. The address generation in FFTs can be simplified by investigating the access patterns at bit-level and results in simple rotations. We have described the address generation units for the FFT processor earlier in [7].

High performance FFT implementation requires parallel memory accesses. We have avoided the multi-port memories due to their high power consumption and used parallel single-port memories instead. Parallel memory organization requires that the operands to be accessed in parallel are distributed to different memory modules, which requires suitable conflict-free access scheme to be used in the parallel memory control logic. In this work, we have used the simple parity scheme proposed in [12].

Another memory related issue in FFTs is the twiddle factor storage. Here we have designed a special unit, which computes the twiddle factors based on complex-valued coefficients stored in a lookup table. Several methods have been proposed to minimize the lookup tables for generating twiddle factors but mainly for radix-2 algorithms. We have developed a method for minimizing the coefficient tables for radix-4 FFT such that also mixed-radix algorithms are supported and such a twiddle factor unit is described in detail in [8]. For N-point FFT, we need to store $N/8+1$ complex-valued coefficients in a table and four real-valued adders and some control logic are needed to produce the twiddle factors. The units can generate twiddle factors for various transform sizes and the maximum supported FFT size depends on the size of the lookup table.

The organization of the proposed transport triggered architecture processor tailored for FFT computations is depicted in Fig. 3. The instruction unit (iu) fetches and decodes the instructions from the instruction memory and generates control signals. The immediate unit is used for extracting immediate data from instructions. The interconnection network consists of one Boolean-valued bus mainly for transporting results from comparison unit to be used with conditional execution and 17 32-bit buses. The network is optimized (not fully connected) and several of the buses are point-to-point connections.

In general, software implementations of FFT contain three nested loops and separate iteration counters are maintained for butterfly index and butterfly column index. Here separate counters are not needed since the twiddle factor generator and address generation units can determine sufficient information from a single

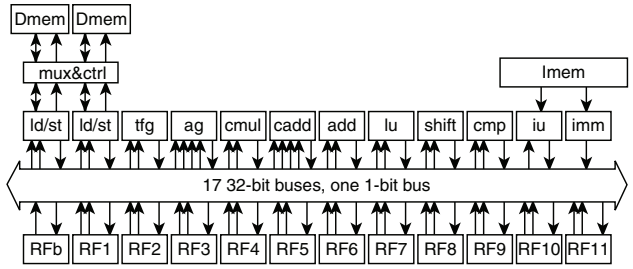

Fig. 3. Block diagram of FFT processor. Dmem: data memory. Imem: instruction memory. ld/st: load-store unit. tfg: twiddle factor unit. ag: address generator. cmul: complex multiplier. cadd: complex adder. add: adder. lu: logical unit. shift: shifter. cmp: compare unit. iu: Instruction unit. imm: immediate unit. RF: register file. RFb: Boolean register file.

iteration counter and transform size register. This results in a single iteration loop implementation as illustrated in the pseudo code in Fig. 4. In addition, the implementation exploits software pipelining and instruction level parallelism. During the kernel execution, one operand read and result write are performed at each cycle, thus the memory bandwidth of the 2-port parallel memory is fully used. The efficiency of the implementation can be considered by noting that theoretical lower bound for 1024-point radix-4 FFT when using a dual-port memory is 5121 memory cycles, i.e., $1024\log_4(1024)+1$ parallel reads and writes to dual-port memory. In the proposed processor with two parallel memory modules as illustrated in Fig. 3, 1024-point FFT requires 5160 cycles. This shows that the software overheads are minimal.

The power consumption due to long instructions is reduced by exploiting dictionary-based code compression [13]. All the unique instructions are stored to a dictionary and replaced with indices pointing to the dictionary. During execution, the code word fetched from the program memory is used to read the original instruction from the dictionary for decoding. The actual dictionary is implemented using standard cells.

4. EXPERIMENTS

We have described processor in VHDL language such that the twiddle factor generator unit supports power-of-two FFTs up to 16K, i.e., the lookup table in the twiddle factor unit contains 2049 complex-valued coefficients. The unit has two pipeline stages and the look up table was implemented as hardwired logic. The native word in the processor is 32 bits, which allows complex-valued data to be represented with 16-bit real part and 16-bit imaginary part. Clock gating was applied to reduce the power consumption of non active function units. This provides savings on units with low utilization.

The design has been synthesized with Synopsys tools onto a 130 nm standard cell technology. Estimates for power consumption are obtained with Synopsys tools with the aid of gate level simulations. The characteristics of the results are listed in Table 1. The twiddle factor unit uses about 23% of the core area and 7% of the power consumption, thus twiddle factor unit improves the energy-efficiency of FFT computations. The most significant power savings compared to our previous results from [7] are due to data memories as here we have exploited two parallel single-port memories instead of dual-port memories which halved the memory power consumption.

The energy-efficiency of FFT implementations is often compared by measuring how many 1024-point FFTs can be computed with energy of 1 mJ, thus we selected some examples from the literature representing FFTs with different implementation technologies. The comparison results are listed in Table 2. The Intel Pentium-4 [14] represents a general-purpose RISC while StrongArm SA-1100 [15] can be considered as a general-purpose processor for mobile devices as it employs custom circuits, clock gating, and reduced supply voltage. Representatives of general-purpose DSP processors are TI TMS320C6416, which is a VLIW machine, and Imagine [16] designed for media applications. Both the processors utilize pseudo-custom data path tiling. In addition, pass-gate multiplexer circuits are exploited in C6416. FFT implementations on C6416 are reported in [17]. It should be noted that cycle count of 6002 is obtained with eight memory ports while the proposed processor uses only two. Spiffee processor [18] is tailored for FFT and power consumption is reduced by using low supply voltages. In [19], an FPGA solution with dedicated embedded FFT logic is reported. In [2], a custom scalable IP core is reported, which employs single memory architecture with clock gating, while in [20] a custom variable-length FFT-processor employing radix-2/4/8 single-path delay algorithm is described. Highly optimized VLSI implementation of FFT using subthreshold circuit techniques is described in [21]. The comparison in Table 2 shows that the energy- efficiency of the proposed processor is in the level of fixed-function ASIC implementations although the implementation is programmable.

```
main() {
   initialization(); /* 8 to 42 instructions */
   prologue(); /* 14 instr.  */
   for(idx=0; idx < (N⌈log4N⌉)/16-1; idx++)
      kernel(); /* 16 instr.  */
   epilogue; /* 17 instr.  */
 }
```

Fig. 4. Pseudo code illustrating structure and control flow of the program code.

Table 1. Characteristics of the proposed processor synthesized on 130nm ASIC technology.

supported FFT sizes	64 – 16384
cycle count	207 – 114722
execution time	828ns – 459μs @ 250MHz
power consumption	60 – 73mW @ 1.5V, 250MHz
max. clock freq.	255 MHz
Area [kgates]	
Core	38
Imem	2
Dmem	240
Total	280
1024-point FFT	
cycle count	5160
power consumption	60.4mW @ 1.5V, 250MHz
	29.8mW @ 1.1V, 140MHz
8192-point FFT	
cycle count	57396
power consumption	68.7mW @ 1.5V, 250MHz

Table 2. Energy-efficiency comparison. V_{CC}: supply voltage. t_{clk}: clock frequency. t_{FFT}: FFT execution time.

Design	Tech. [nm]	V_{CC} [V]	t_{clk} [MHz]	t_{FFT} [μs]	FFT/mJ
1024-point FFT					
proposed	130	1.5	250	21	809
	130	1.1	140	37	910
[19]	130	1.3	100	13	149
	130	1.3	275	5	241
[17]	130	1.2	720	8	100
	130	1.2	300	22	250
[14]	130	1.2	3000	24	1
[16]	150	1.5	232	160	16
[21]	180	0.9	6	430	1428
	180	0.35	0.01	250000	6452
[2]	180	–	20	282	43
[20]	350	3.3	45	23	93
	350	2.3	18	57	133
[15]	350	2.0	74	426	60
[18]	600	3.3	173	105	39
	600	1.1	16	330	319
8192-point FFT					
proposed	130	1.5	250	230	63
[22]	180	1.8	20	717	55
[1]	180	–	22	908	35
[5]	250	–	12	1198	4

5. CONCLUSIONS

In this paper, a low-power application-specific processor tailored for FFT computation was proposed where several methods for reducing the power consumption were utilized: special function units, parallel memories, clock gating, and code compression. The processor was synthesized on a 130 nm ASIC technology and power analysis showed that the proposed processor has both high energy-efficiency without significant reduction on performance. The performance can even be improved by adding computational resources.

6. REFERENCES

[1] C.-L. Wey, S.-Y. Lin, W.-C. Tang, and M.-T. Shiue, "High-speed, low cost parallel memory-based FFT processors for OFDM applications," in *IEEE Int. Conf. Electronics Circ. Syst.*, Marrakech, Marocco, Dec. 11–14 2007, pp. 783–787.

[2] Y. Zhao, A. T. Erdogan, and T. Arslan, "A low-power and domain-specific reconfigurable FFT fabric for system-on-chip applications," in *Proc. IEEE Par. Distributed Process. Symp. Reconf. Logic*, Denver, CO, Apr. 4–8 2005.

[3] W. Han, A. T. Erdogan, T. Arslan, and M. Hasan, "High-performance low-power FFT cores," *ETRI Journal*, vol. 30, no. 3, pp. 451–460, June 2008.

[4] C.-H. Hung, S.-G. Chen, and K.-L. Chen, "Design of an efficient variable-length FFT processor," in *Proc. IEEE ISCAS*, Vancouver, Canada, May 23–26 2004, vol. 2, pp. 833–836.

[5] X. Li, Z. Lai, and J. Cui, "A low-power and small area FFT processor for OFDM demodulator," *IEEE T. Consumer Electr.*, vol. 53, no. 2, pp. 274–277, May 2007.

[6] G. Liu and Q. Feng, "ASIC design of low-power reconfigurable FFT processor," in *Int. Conf. ASIC*, Guilin, China, Oct. 22–25 2007, pp. 44–47.

[7] T. Pitkänen, R. Mäkinen, J. Heikkinen, T. Partanen, and J. Takala, "Transport triggered architecture processor for mixed-radix FFT," in *Conf. Record Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, Oct. 29–Nov. 1 2006, pp. 84–88.

[8] T. Pitkänen, T. Partanen, and J. Takala, "Low-power twiddle factor unit for FFT computation," in *Embedded Computer Systems: Architectures, Modeling, and Simulation: Proc. Int. Workshop SAMOS 2007*, S. Vassiliadis, M. Bereković, and T. D. Hämäläinen, Eds., Berlin, Germany, 2007, vol. LNCS 4599, pp. 273–282, Springer-Verlag.

[9] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1975.

[10] J. Granata, M. Conner, and R. Tolimieri, "Recursive fast algorithms and the role of the tensor product," *IEEE Trans. Signal Processing*, vol. 40, no. 12, pp. 2921–2930, Dec. 1992.

[11] H. Corporaal, *Microprocessor Architectures: From VLIW to TTA*, John Wiley & Sons, Chichester, UK, 1997.

[12] D. Cohen, "Simplified control of FFT hardware," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 24, no. 6, pp. 577–579, Dec. 1976.

[13] J. Heikkinen, *Design and Analysis of Program Compression Methods for Customizable Parallel Processor Architectures*, Ph.D. thesis, Tampere Univ. Tech., Finland, 2007.

[14] M. Deleganes, J. Douglas, B. Kommandur, and M. Patyra, "Designing a 3 GHz, 130 nm, Intel® Pentium ®4 processor," in *Digest Technical Papers Symp. VLSI Circuits*, Honolulu, HI, June 13–15 2002, pp. 230–233.

[15] Intel, *StrongARM SA-110 Microprocessor for Portable Applications Brief Datasheet*, 1999.

[16] S. Rixner, W. J. Dally, U. J. Kapasi, B. Khailany, A. Lopez-Lagunas, P. R. Mattson, and J. D. Owens, "A bandwidth-efficient architecture for media processing," in *Proc. Ann. ACM/IEEE Int. Symp. Microarchitecture*, Dallas, TX, Nov. 30–Dec. 2 1998, pp. 3–13.

[17] Texas Instruments, Inc., Dallas, TX, *TMS320C64x DSP Library Programmer's Reference*, Oct. 2003.

[18] B. M. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE J. Solid State Circuits*, vol. 43, no. 3, pp. 380–387, March 1999.

[19] S. Y. Lim and A. Crosland, "Implementing FFT in an FPGA co-processor," in *Proc. Int. Embedded Solutions Event*, Santa Clara, CA, 2004, pp. 230–233.

[20] Y.-T. Lin, P.-Y. Tsai, and T.-D. Chiueh, "Low-power variable-length fast Fourier transform processor," *Proc. IEE Comput. Digital Techniques*, vol. 152, no. 34, pp. 499–506, July 8 2005.

[21] A. Wang and A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE J. Solid State Circuits*, vol. 40, no. 1, pp. 310–319, Jan. 2005.

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