also would like to thank the reviewers for their comments to improve the quality of this publication.

REFERENCES

[1] J. Crols and M. Steyaert, "Switched opamp: An approach to realize full-CMOS switched-capacitor circuits at very low power supply voltages," *IEEE J. Solid-State Circuits*, vol. 29, pp. 936–942, Aug. 1994.
[2] F. Maloberti, F. Francesconi, P. Malcovati, and O. Nys, "Design consideration on low-voltage low-power data converters," *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 853–863, Nov. 1995.
[3] R. Castello, F. Montecchi, F. Rezzi, and A. Baschirotto, "Low-voltage analog filters," *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 827–840, Nov. 1995.
[4] J. Grilo, E. MacRobbie, R. Halim, and G. C. Temes, "1.8 V, 94 dB dynamic range DE modulator for voice applications," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 1996, pp. 230–231.
[5] V. Peluso, P. Vancorenland, A. Marques, M. Steyaert, and W. Sansen, "A 900-mW low-power $\Delta\Sigma$ A/D converter with 77-dB dynamic range," *IEEE J. Solid-State Circuits*, vol. 33, pp. 1887–1897, Dec. 1998.
[6] M. Dessouky and A. Kaiser, "Very low-voltage digital-audio $\Delta\Sigma$ modulator with 88-dB dynamic range using local switch boothstrapping," *IEEE J. Solid-State Circuits*, vol. 36, pp. 349–355, Mar. 2001.
[7] M. Keskin, "Low voltage switched capacitor circuits for lowpass and bandpass $\Delta\Sigma$ converters," Ph.D. dissertation, Oregon State Univ., Corvallis, OR, Jan. 2002.
[8] M. Keskin, U. Moon, and G. C. Temes, "A 1-V, 10-MHz clock-rate, 13-bit CMOS $\Delta\Sigma$ modulator using unity-gain-reset opamps," *IEEE J. Solid-State Circuits*, vol. 37, pp. 817–824, July 2002.
[9] S. Karthikeyan, S. Mortezapour, A. Tammineedi, and E. K. F. Lee, "Low-voltage analog circuit design based on inverting opamp configuration," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 176–184, Mar. 2000.
[10] S. Karthikeyan, A. Tammineedi, C. Boecker, and E. K. F. Lee, "Design of Low-voltage front-end interface for switched-opamp circuits," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 176–184, Mar. 2000.
[11] A. Baschirotto, "A low-voltage sample-and-hold circuit in standard CMOS technology operating at 40 MS/s," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 394–399, Apr. 2001.
[12] D. Chang and U. Moon, "1-V input sampling circuit with improved linearity," *Electron. Lett.*, vol. 37, no. 8, pp. 479–481, Apr. 2001.

Efficient FPGA-Implementation of Two's Complement Digit-Serial/Parallel Multipliers

Javier Valls and Eduardo Boemo

***Abstract*—This paper presents an efficient implementation of digit-serial/parallel multipliers on 4-input look-up table (LUT)-based field programmable gate arrays (FPGAs). This subset of FPGA devices hide individual gate delays and add important wiring delay. These two facts produce important changes over the theoretical advantages of each topology. Architectural transformations are applied to obtain topologies with minimum logic depth and where the maximum clock speed is limited by the FPGA technology. The main results of applying those transformations to the different multipliers have been quantified for Altera FLEX10K family, and the conclusions have been extrapolated to other FPGA families.**

***Index Terms*—Digit-serial arithmetic, field programmable gate arrays (FPGAs), serial/parallel multiplier.**

I. INTRODUCTION

Real-time signal processing hardware requires efficient multiplier units. However, each application demands a different sample rate. From speech to image or radar, a wide frequency range is required. In most of the technologies, a bit-parallel circuit is expensive: its cost in area is critical, and runs faster than the throughput required by the application. At this point, the bit-serial [1], [2], [3] or digit-serial [4], [5], [6], [7], [8] approaches become an important alternative. Furthermore, in FPGA-targeted applications, the serial stream of data matches better with the structure of such devices [9].

This paper presents a systematic study about the FPGA-implementation of digit-serial/parallel multipliers. In Section II different multiplier topologies are presented. In Section III, actual results are summarized, and different techniques to enhance the performance are evaluated. All the analyzed circuits have been implemented on an EPF10K50GC403-3 FPGA [10]. These prototypes compute 8-bit two's complemented words (data and coefficients), and digit-sizes of $N = 1$, 2, 4, and 8 bits. For clarity, the experimental part is divided into the following subsections. Section III-A reviews the conventional methods to pipeline serial/parallel multipliers (SPMs), and demonstrates that this technique results in inefficiency in some of the circuits. In these cases, a new alternative to pipelining is also proposed. In Section III-B, a mechanism of asynchronous clear or set of the registers is presented. It reduces the area of some bit-serial multiplier versions. Finally, a modified structure that diminishes the logic depth is explained in Section III-C.

II. SERIAL/PARALLEL MULTIPLIERS (SPMS)

SPMs are embedded in digital signal processing (DSP) blocks to compute the multiplication of a coefficient by data. The coefficient is expressed in parallel form while the data enters to the multiplier as a serial stream.

Manuscript received July 28, 2000; revised February 10, 2003. This work was supported by the Spanish Ministry of Science and Technology, under Contract TIC2001-2688-02 and Contract TIC2001-2688-C03-03. This paper was recommended by Associate Editor Y. Wang.

J. Valls is with E.P.S. GANDIA, Departamento Ingenieria Electronica, Universidad Politecnica de Valencia, 46730 Gandia (Valencia), Spain (e-mail: jvalls@eln.upv.es).

E. Boemo is with the School of Computer Engineering, Universidad Autonoma de Madrid, 28049 Madrid, Spain (e-mail: eduardo.boemo@uam.es).

Digital Object Identifier 10.1109/TCSII.2003.811438

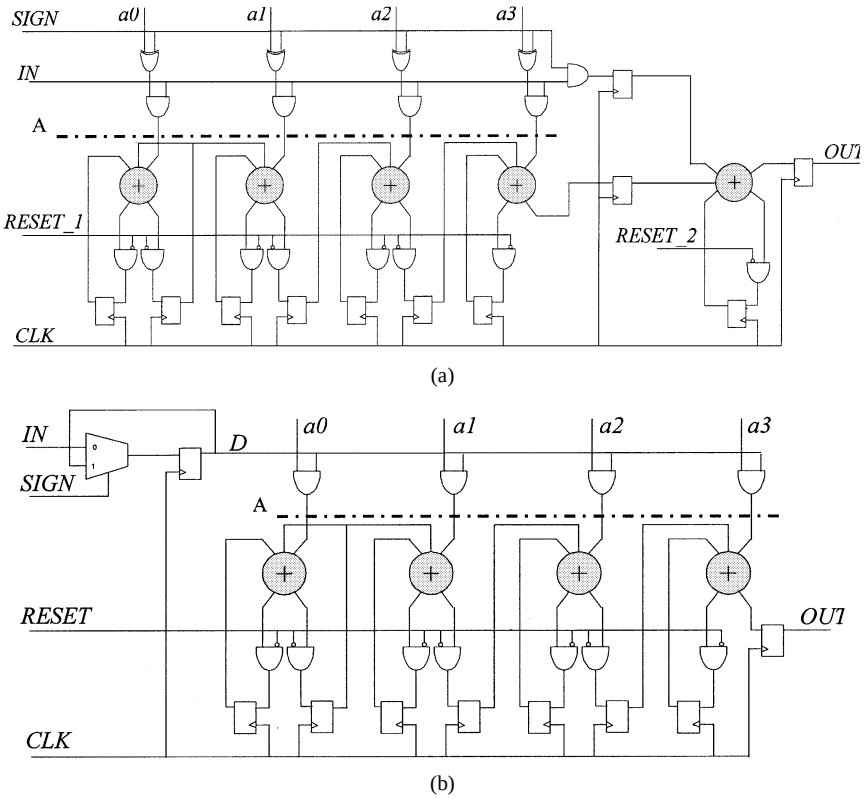

Fig. 1. Bit-serial multipliers. (a) SPM-I. (b) SPM-II.

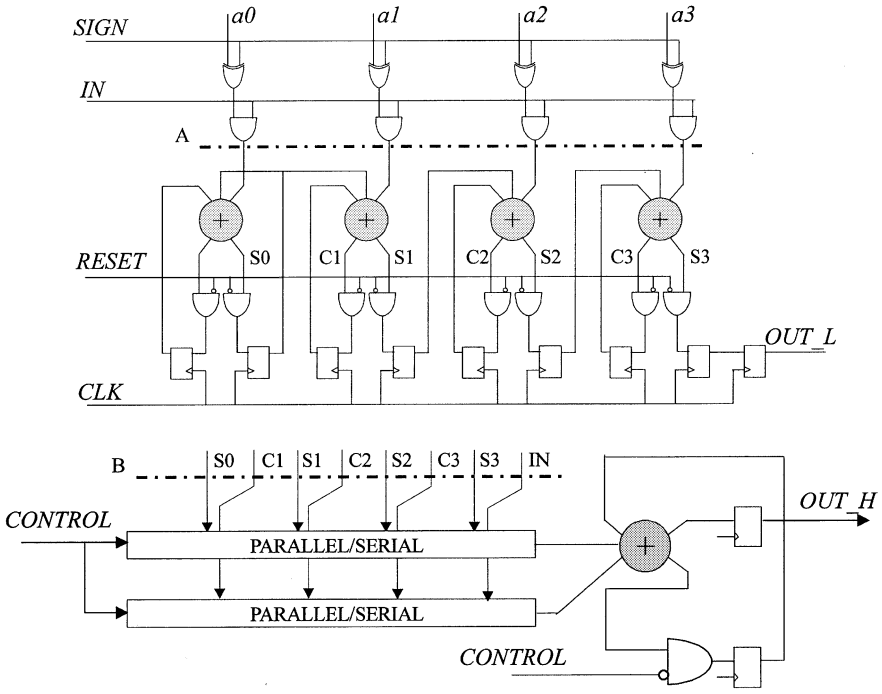

Fig. 2. Bit-serial DPSPM.

From the FPGA implementation point of view, where logic is mapped into look-up tables (LUTs), there are two alternatives to compute single precision serial/parallel multiplication of two's complement numbers. The main difference between them is the way to process the sign-bit of the input data. In this paper, these two alternatives are named SPM-I and SPM-II. The first does not extend the sign bit of the input data [4], [11], [12], [13] while the second does [4], [5], [14]–[18]. Their bit-serial circuits are shown in Fig. 1.

The computational scheme of SPM-I can be used to design the double precision SPMs (DPSPMs) [6], [7], [19]. A bit-serial DPSPM is depicted in Fig. 2. The goal of this circuit is to maintain the throughput without either adding the extra clock cycles to insert zeros, or extending the sign-bit.

TABLE I
RESULTS OF THE FPGA-IMPLEMENTATION

Circuit	N (bits)	f_c (MHz)	T_{pro} (ns)	f_s (MHz)	A (LEs)	LD (LEs)	Ef. AxT	Ef. A_TxT
SPM-I	1	105.3	7.2	6.6	35	2	4.99	8.41
	2	95.2	8.2	11.9	53	2	4.45	6.47
	4	53.2	16.5	13.3	87	4	6.54	8.35
	8	26.2	35.9	13.1	157	8	11.85	13.68
SPM-II	1	125	3.4	8.3	33	2	3.96	6.84
	2	88.5	9	11.1	50	2	4.52	6.69
	4	54.3	16.1	13.6	84	4	6.18	7.95
	8	26.2	35.8	13.1	153	8	11.66	13.49
DPSPM	1	57.1	15.2	7.1	64	3	8.96	12.32
	2	63.7	13.4	15.9	80	3	5.02	6.53
	4	44.2	20.3	22.1	111	5	5.02	6.10
	8	25.5	36.9	25.5	187	9	7.33	8.27

III. IMPLEMENTATION: MAPPING ON $K = 4$ LUT-BASED FPGAS

The digit-serial/parallel multipliers presented in previous section were implemented using an EPF10K50-3 FPGA [10]. Each multiplier family uses 8-bit data and coefficients. Their versions include digit-sizes of $N = 1$, 2, 4, and 8 bits. The place and route of the circuits was performed using the default options of the tools, excepting the indication of the wires that require fast carry lines [10]. Every circuit version was evaluated according to the following parameters: maximum clock frequency (f_c); maximum propagation delay ($T_{\rm pro}$); maximum sample frequency (f_s); area (A); logic depth (LD); and finally, the area-time product ($A \times T$). In Table I, the implementation results of the SPM-I, SPM-II, and DPSPM are presented, and the first area-time figure can be obtained.

It is important to remark that some of the theoretical advantages of DPSPM circuits are hidden by an FPGA implementation, the technological framework selected in this work. For instance, theoretically the clock frequency of both SPM and DPSPM for the same digit-size should be identical. Thus DPSPMs should double the sample rate with respect to SPMs (because the former do not require to insert extra zeroes). However, in the implemented versions, SPMs run faster than DPSPMs. Thus the resulting sample rates of DPSPMs are just only a little bit higher (see Table I). This effect is a consequence of the fixed structure of the selected FPGA: a matrix organization in which each element is a 4-input LUT. So, the circuits have to be divided into 4-input functions in order to be implemented. The logic depth of SPM and DPSPM is 2 LUTs and 3 LUTs, respectively, as can be seen in Fig. 3. The logic depth increment in the DPSPM is caused by the extra PSCs. As a result, a throughput degradation with respect to the ideal case is produced.

Partitioning logic into LUTs also causes that both bit-serial circuits and $N = 2$ bits digit-size ones have the same logic depth. As a consequence, both versions could ideally achieve the same clock frequency. Table III gives an example of this effect: $N = 2$ DPSPM achieves higher clock rates than bit-serial one.

Considering that FPGA-vendors are permanently marketing chips with different LUT-size (named k), the optimal value of k that will allow this kind of topology to achieve a logic depth reduction is summarized in Table II for different versions of the circuit. The k value in the k-*LUT* column can be reduced in one unit for SPM-I and SPM-II circuits, if the device also incorporates dedicated logic to implement the synchronous reset of flip-flops.

The case of Xilinx devices whose configurable logic elements (CLBs) consist of 4-input LUTs is a little bit different. On the one hand, FPGA families like Spartan II, Virtex, and Virtex II contain dedicated logic to perform synchronous reset of the flip-flop. On the other hand, these devices include dedicated multiplexors (called MUXFx) that allow combining several 4-input LUTs to implement functions with higher number of inputs inside a CLB. The XC4000

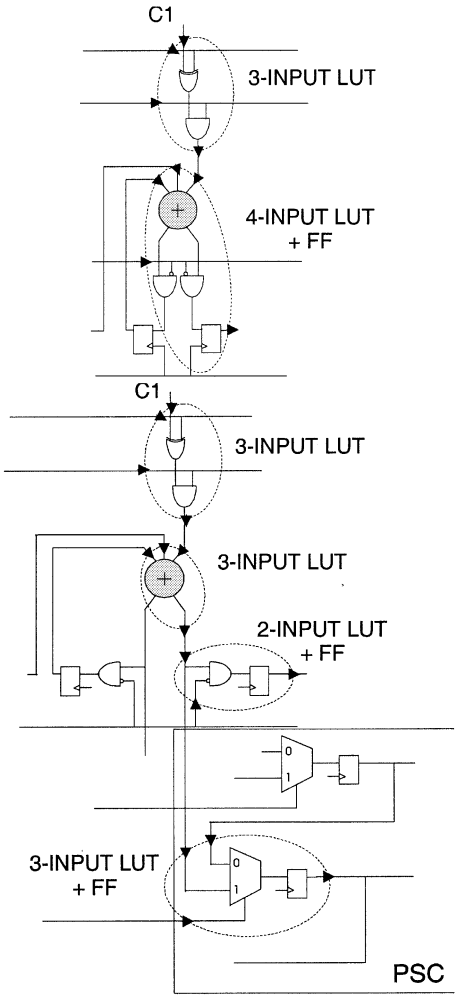

Fig. 3. Multipliers cell separated in LEs: (a) SPM-I; (b) DPSPM.

TABLE II
REQUIRED LUT-SIZE TO ACHIEVE A LOGIC DEPTH REDUCTION

Multiplier	Digit-size (N)	k-LUT	LD (LEs)
SPM-I	1	6	1
SPM-II	1	5	1
DPSPM	1	7	1
SPM-I	2	12	1
SPM-II	2	11	1
DPSPM	2	14	2

and Spartan devices can implement functions up to five inputs in one CLB, Spartan II and Virtex up to six inputs in one CLB, and Virtex II up to seven inputs in one CLB and eight inputs using two CLBs. Hence, bit-serial SPM-I and SPM-II multipliers will achieve minimum logic depth in such devices. Finally, pointing out that although newer Altera families (Apex and Mercury) also include the logic resources to perform the synchronous reset, it cannot be used to reduce the logic depth of the target circuits because these hardware resources cannot be used when LEs are configured in normal mode (that only can be used as one 4-input LUT), but in counter mode (configured as two 3-input functions within a LE).

A. Pipelining

The feedback loops present in the serial/parallel multipliers limit the application of pipelining: it can only be performed by registering the

TABLE III
RESULTS OF THE FPGA-IMPLEMENTATION OF THE PIPELINED MULTIPLIERS

Circuits	Pipeline cutsets	N (bits)	f_c (MHz)	T_{pro} (ns)	f_s (MHz)	A (LEs)	LD (LEs)	Ef. AxT	Ef. A_TxT
SPM-I	A (Fig.1a)	1	125	3,7	8,3	36	1	4,32	7,20
		2	99	7,8	12,4	62	2	51	6,95
		4	49.8	17,8	12,4	110	4	8,84	10,77
SPM-II	A (Fig.1b)	1	125	3,5	8,3	33	1	3,96	6,84
		2	98	7,9	12,3	58	2	4,73	6,69
		4	49,3	18	12,3	107	4	8,69	10,64
DPSPM	A (Fig.2)	1	69	8,4	11,7	65	2	5,56	7,62
		2	56,5	15,4	14,1	91	3	6,44	8,14
		4	42,6	21,2	21,3	135	5	6,35	7,47
	B (Fig.2)	1	98	7,9	12,3	69	2	5,63	7,59
		2	83,3	9,7	20,8	89	2	4,27	5,42
		4	51,5	17,1	25,8	124	4	4,81	5,74
		8	26,9	34,9	26,9	196	8	7,29	8,18
	Both A and B	1	125	4,7	15,6	69	1	4,42	5,95
		2	90,1	8,8	22,5	97	2	4,31	5,37

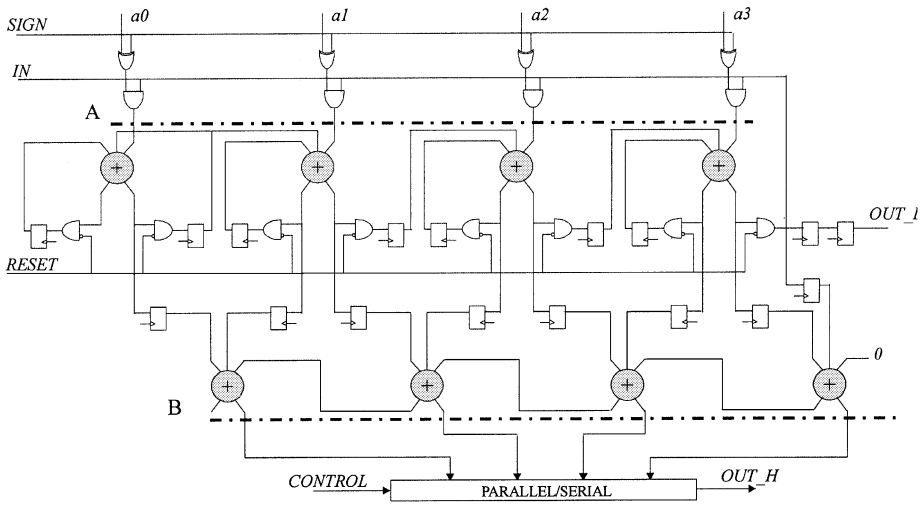

Fig. 4. Modified bit-serial DPSPM.

outputs of the partial-product generator block [20]. The implementation results of pipelining the previous circuits are shown in Table III. In most of the cases, the throughput is not improved with respect to the original versions. Bit-serial circuits ($N = 1$ bit) are the exception: they reduce the logic depth in one LUT, with no area penalization. On the contrary, pipelined DSMs ($N > 1$ bit), are larger and do not exhibit a logic depth reduction. This situation is repeated in every double precision multiplier version. For example, for $N = 2$ bits, an FPGA with 9-input LUTs would be necessary to get a speedup. When pipelining is applied in both A and B cutsets, bit-serial DPSPM circuits reaches the maximum clock frequency of the chip. The area is only incremented in 5 LEs. As final remark, the technique only is suitable for $N = 1$, where an effective logic depth reduction is achieved.

If pipelining were applied to implement these circuits on Xilinx devices we need not use the A cutset in bit-serial SPM-I and SPM-II topologies, because they already exhibit minimum logic depth (1 CLB). Furthermore, the A cutset would be required to achieve the minimum logic depth in bit-serial DPSPM. As in the case of Altera devices, it does not lead to any advantage to pipeline digit-serial versions of the multipliers, the logic depth remains constant.

B. Asynchronous CLEAR of FFs

The set and clear of the flip-flops (FFs) in serial/parallel multiplier implementations is conventionally performed asynchronously [14], [19], [16]. Considering that most commercial FPGAs include an asynchronous clear and set, this feature can be utilized to eliminate one input signal of each LUT (the *RESET* signal) in those devices that do not incorporate dedicated resources to perform the synchronous reset (XC4000 and Spartan of Xilinx and FLEX8K and FLEX10K of Altera). In this way, a logic depth reduction can be obtained, at the cost of one extra clock cycle to compute each word. The final balance between the potential speedup caused by a lower logic depth (and its corresponding wiring reduction), and the extra delay introduced by these additional clock cycles will depend on the chip model utilized to build the circuit.

From the previous topologies, only the bit-serial SPM-II can take full advantage of this idea. In this circuit, each slice consists of two 5-input functions and 2 FFs. It can be mapped using 3 LEs, having a logic depth of 2 LUTs. By using an asynchronous clear, each cell requires two 4-input functions (2 LEs), reducing the logic depth to just one LUT. The experimental results indicate that both logic depth and area requirements decrease (see Table IV). Nevertheless, there is not an effective speed increment for the selected chip. The parameter $T_{\rm pro}$ has been reduced, but the saturation frequency (125 MHz) has been reached.

The previous optimization cannot be obtained in the other circuits: its applicability will depend on the FPGA architecture. For example, the bit-serial SPM-I could be optimized if 5-input LUT FPGAs were available (the case of Xilinx FPGAs), meanwhile the 2-bit digit-size SPM-I and SPM-II would require 10-input and 9-input LUTs respectively to take advantage of this method.

TABLE IV
BIT-SERIAL SPM-II WITH ASYNCHRONOUS CLEAR OF THE FF

Circuit	N (bits)	f_c (MHz)	T_{pro} (ns)	f_s (MHz)	A (LEs)	LD(Les)	Ef. AxT	Ef. A_TxT
SPM-II	1	125	1,6	7,8	25	1	3,07	6,14

TABLE V
FPGA-IMPLEMENTATION OF THE MDPSPM

N	f_c (MHz)	T_{pro} (ns)	f_s (MHz)	A (LEs)	LD (LEs)	Ef. AxT	Ef. A_TxT
1	89,3	8,9	11,2	64	8^*+1	5,73	7,89
2	87,7	9,1	21,9	81	8^*+1	3,69	4,79
4	49	18,1	24,5	112	4	4,57	5,22
8	28,8	32,4	28,8	169	8	5,87	6,70

TABLE VI
FPGA-IMPLEMENTATION OF THE PIPELINED MDPSPM

Pipeline cutsets	N	f_c (MHz)	T_{pro} (ns)	f_s (MHz)	A (LEs)	LD(Les)	Ef. AxT	Ef. A_TxT
A	1	91.7	8.6	11.5	65	8^*+1	5.69	7.76
	2	87	9.2	21.7	88	8^*+1	4.09	5.20
B	1	81,3	10	10,2	66	2	6,49	8,86
	2	81,3	10	20,3	83	2	4,08	5,26
Both A and B	1	125	4,5	15,6	67	8^*	4,29	5,82
	2	92,6	8,5	23,1	92	2	3,98	5,01

C. Modification of the DPSPM (MDPSPM)

The modified multiplier structure presented in [21] is based on the fast SPM proposed by R. Gnanasekaran [22]. The main idea is to avoid the W extra clock cycles required to complete the computation, by including a bit-parallel adder. Thus, the sum and carry vectors are computed in parallel after the first W cycles. The adder block replaces the W clock cycles needed to achieve the same operation serially. The DPSPM circuit proposed in [22] can be transformed in a double precision serial/parallel multiplier, simply by adding a PSC to the bit-parallel outputs of the RCA (Ripple Carry Adder). Commercial FPGAs usually allow the designer to build fast and small ripple-carry adders by using especial carry-chain lines [10]. Then, by modifying the circuit in such a way, a logic depth reduction can be achieved. These multipliers can be directly replaced by the MDPSPMs, without any circuit modification. The bit-serial version of this multiplier is shown in Fig. 4.

The results of the FPGA implementation are reported in Table V. The only difference respect to the previous circuits is that the FAST logic synthesis option (the assignation of carry chain lines) has been used to map the bit-parallel adder. Thus, the MDPSPM topology achieves the best performance (Table V). The speedup goes from 1.56 (for bit-serial version) to 1.15 (for bit-parallel version). The modified circuits have the same logic depth than the SPM ones. This is true for every digit-serial version, but they are one LUT smaller than the corresponding DP-SPMs. As a result, they achieve a higher throughput than the conventional DPSPM with nearly the same cost in area. If MDPSPM are compared to the single precision circuits, the throughput improvement varies from 1.35 to 1.69 (for the bit-serial version), up to 2 (for the bit-parallel version). Once again, the enhancement is obtained without incrementing the latency. The MDPSPM circuits lead to several changes in the optimized area-time figure of the multipliers.

In Table VI, the results of pipelining the MDPSPM are presented. As was remarked in Section III-A, pipelining the circuit after the partial product generation (cutset A in Fig. 4) increases the area by increments but does not reduce the logic depth. As a consequence, only the results for $N = 1$ and 2 bits are useful for custom DSP designers.

In the MDPSPM, pipelining can be extended to the RCA outputs (cutset B in Fig. 4). Results for the three pipeline alternatives (A, B, and both A and B cutsets) are reported in Table VI.

The main result can be summarized as follows: by pipelining in point A and B, the bit-serial MDPSPM reaches the maximum frequency imposed by the process technology (125 MHz). The cost in area is minimum (just 3 extra LEs), but the penalty to be paid is an increase in latency of two more cycles. This structure modifies the area-time figure in the range 12 MHz to 15.5 MHz, saving 14 LE's (17%) with respect to the previous alternative.

Finally, it is important to note that this modification should not be applied to Xilinx FPGAs: it only adds an enhancement in bit-serial and 2-bit digit-size multipliers, and these circuits directly achieve the minimum logic depth (maximum speed) or can easily achieve it by pipelining, as shown in previous section.

IV. CONCLUSIONS

This work presented a systematic study of the FPGA-implementation of digit-serial/parallel multipliers. The target technology has been a $k = 4$ LUT-based FPGA, but optimal results have been extended to other LUT sizes. Three types of serial/parallel multipliers (two of single precision, and one of double precision) have been evaluated. Pipelining has been applied to extend the speed of each class of multiplier. Several methods have been proposed to obtain a logic depth reduction, obtaining the following conclusion:

Conventional pipelining only leads to a logic depth reduction in bit-serial circuits. It is not a suitable technique in digit-serial SPM and DPSPM circuits. Minimum logic depth is achieved in bit-serial DPSPM if an extra pipelining (cutset B) is applied.

In the bit-serial SPM-II, the asynchronous clear of the FFs reduces the logic depth in one LUT and the area in W LEs.

The proposed modification of the DPSPM improves the performance for moderate word-lengths.

REFERENCES

[1] L. B. Jackson, J. F. Kaiser, and H. S. McDonald, "An approach to the implementation of digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 413–421, 1968.

[2] R. F. Lyon, "A bit-serial VLSI architectural methodology for signal processing," in *1st Int. Conf. on Very Large Scale Integration*, J. P. Gray, Ed., 1981, pp. 131–140.

[3] P. Denyer and D. Renshaw, *VLSI SIGNAL PROCESSING: A Bit-Serial Approach*. Reading, MA: Addison-Wesley, 1985.

[4] S. G. Smith and P. B. Denyer, *Serial Data Computation*. Boston, MA: Kluwer Academic, 1988.

[5] K. K. Parhi, "A systematic approach for design of digit-serial signal processing architectures," *IEEE Trans. Circuits and Systems*, vol. 38, pp. 358–375, Apr. 1991.

[6] R. Hartley and P. Corbett, "Digit-serial processing techniques," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 707–719, June 1990.

[7] R. I. Hartley and K. K. Parhi, *Digit-Serial Computation*. Boston, MA: Kluwer Academic, 1995.

[8] J. Valls, M. M. Peiro, T. Sansaloni, and E. Boemo, "A study about FPGA-based digital filters," in *1998 IEEE Workshop on VLSI Signal Processing: Design and Implementation (SiPS'98)*, Boston, MA, pp. 192–201.

[9] R. Petersen and B. Hutchings, "An assessment of the suitability of FPGA-based systems for use in DSPs," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1995, pp. 293–302.

[10] *Altera Data Book*, San Jose, CA, 2000.

[11] G. P. Alexiou and N. Kanopoulos, "A new serial/parallel two's complement multiplier for VLSI digital signal processing," *Int. J. Circuit Theory Applicat.*, vol. 20, no. 2, pp. 209–214, 1992.

[12] A. Bermak, D. Martinez, and J.-L. Noullet, "High-density 16/8/4-bit configurable multiplier," *Proc. IEE Circuits, Devices & Systems*, vol. 144, no. 5, pp. 272–276, 1997.

[13] P. Larsson-Edefors, "A 965-Mb/s 1.0-μm standard CMOS twin-pipe serial/parallel multiplier," *IEEE J. Solid-State Circuits*, vol. 31, Feb. 1996.

[14] S. A. White, "Digital adaptive-element building blocks for MOS large-scale integration," *IEEE Trans. Computers*, vol. C-18, pp. 699–706, Aug. 1969.

[15] D. J. Myers and P. A. Ivey, "Circuits elements for VLSI signal processing," *Br. Telecom Technol. J.*, vol. 2, pp. 67–77, July 1984.

[16] L. Wanhammar, *DSP Integrated Circuits*. New York: Academic, 1999.

[17] L. Dadda, "On serial-input multipliers for two's complement numbers," *IEEE Trans. Computers*, vol. 38, Sept. 1989.
[18] P. Ienne and M. A. Viredaz, "Bit-serial multipliers and squarers," *IEEE Trans. Computers*, vol. 43, Dec. 1994.
[19] P. Ingelhan, B. Jonsson, B. Sikstrom, and L. Wanhammar, "A high-speed bit-serial processing element," in *Proc. Eur. Conf. on Circuit Theory and Design (ECCTD'89)*, 1989, pp. 162–165.
[20] S. G. Smith, "Serial/parallel automultiplier," *Electron. Lett.*, vol. 23, no. 8, pp. 413–415, 1987.
[21] J. Valls, T. Sansaloni, M. M. Peiró, and E. Boemo, "Fast FPGA-based pipelined digit-serial/parallel multipliers," in *IEEE Int. Symp. on Circuits and Systems (ISCAS'99)*, vol. 1, Orlando, FL, 1999, pp. 482–485.
[22] R. Gnanasekaran, "A fast serial-parallel binary multiplier," *IEEE Trans. Computers*, vol. C-34, Aug. 1985.

An Efficient Pipelined FFT Architecture

Yun-Nan Chang and Keshab K. Parhi

***Abstract*—This paper presents an efficient VLSI architecture of the pipeline fast Fourier transform (FFT) processor based on radix-4 decimation-in-time algorithm with the use of digit-serial arithmetic units. By combining both the feedforward and feedback commutator schemes, the proposed architecture can not only achieve nearly 100% hardware utilization, but also require much less memory compared with the previous digit-serial FFT processors. Furthermore, in FFT processors, several modules of ROM are required for the storage of twiddle factors. By exploiting the redundancy of the factors, the overall ROM size can be effectively reduced by a factor of 2.**

***Index Terms*—Digit-serial, fast Fourier transform (FFT), pipelined FFT, radix-4 FFT.**

I. Introduction

The fast Fourier transform (FFT) plays an important role in the design and implementation of discrete-time signal processing algorithms and systems. In recent years, motivated by the emerging applications in the modern digital communication systems and television terrestrial broadcasting systems, there has been tremendous growth in the design of high-performance dedicated FFT processors [1], [2]. Pipelined FFT processor is a class of real-time FFT architectures characterized by continuous processing of the input data which, for the reason of the transmission economy, usually arrives in the word sequential format. However, the FFT operation is very communication intensive which calls for spatially global interconnection. Therefore, much effort on the design of FFT processors focuses on how to efficiently map the FFT algorithm to the hardware to accommodate the serial input for computation. This paper presents a novel FFT implementation based on the use of digit-serial arithmetic which can lead to very efficient architectures.

Manuscript received August 15, 2000; revised February 12, 2003. This paper was recommended by Associate Editor Y. Wang.

Y.-N. Chang is with the Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan 804, R.O.C. (e-mail: ynchang@cse.nsysu.edu.tw).

K. K. Parhi is with the Department of Electrical and Computer Engineering, University of Minnesota, MN 55455 USA.

Digital Object Identifier 10.1109/TCSII.2003.811439

II. Review of FFT Processors

The discrete Fourier transform (DFT) $X(k)$ of an N-point sequence $x(n)$ is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \qquad k = 0, 1, \ldots, N-1$$
$$W_N = e^{-j(2\pi/N)}. \tag{1}$$

Instead of direct implementation of the computationally intensive DFT, the FFT algorithm is used to factorize a large point DFT recursively into many small point DFTs such that the overall operations involved can be drastically reduced. There are two well-known types of FFT algorithms called *decimation-in-time (DIT)* and *decimation-in-frequency (DIF)* FFT which can be derived from each other by transposition. For example, according to radix-4 *DIT* FFT, (1) can be decomposed and expressed in the matrix form as follows:

$$\left[X(k), X\left(k+\frac{N}{4}\right), X\left(k+\frac{N}{2}\right), X\left(k+\frac{3N}{4}\right)\right]^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^0 F_0(k) \\ W_N^k F_1(k) \\ W_N^{2k} F_2(k) \\ W_N^{3k} F_3(k) \end{bmatrix}. \tag{2}$$

Here

$$F_{n_1}(k) = \sum_{n_2=0}^{(N/4)-1} x(n_1 + 4n_2) W_{N/4}^{n_2 \cdot k}$$
$$\text{for } n_1 = 0, 1, 2, 3;\ k = 0, 1, \ldots, \frac{N}{4} - 1.$$

Radix-2 and radix-4 are the most common radices used in FFT decompositions. Radix-4 decomposition is more attractive since it requires less amount of multiplication operations for FFT and reduces the number of multiplications from N^2 for direct implementation of DFT to only $(\log_4 N - 1)N$.

Since the data sequence $x(n)$ arrives sequentially, the parallel data flow graph has to be projected along the order of input sequence in order to obtain efficient pipeline architectures. As (2) shows, each stage of FFT computation consists of retrieving the data $F_0(k)$, $F_1(k)$, $F_2(k)$, $F_3(k)$ for specific k, and the corresponding twiddle factor multiplication, followed by the multiplication of the radix-4 butterfly matrix. Direct implementation of (2) requires three multipliers to perform the twiddle factor multiplication as shown in Fig. 1(a) [1], [3]. Here, the commutator is used to generate the proper data sequence for the following twiddle factor multiplication by swapping/exchanging the output data coming from the previous stage. The salient feature of this feedforward approach is that the trivial factor $W_N^0 (=1)$ in the twiddle matrix can be reflected in the hardware. However, unless four input data are sampled in parallel, this architecture cannot achieve full efficiency. For most of the applications where FFT processor must be interfaced to a continuous word serial stream, it is only possible to achieve 25% hardware utilization as there is a $4:1$ mismatch between the bandwidth of input data rate and that of the processor. (In general, the utilization for radix-r butterfly unit is $1/r$.) In order to compensate this mismatch, a fully utilized architecture based on the use of digit-serial arithmetic units has been proposed in [4].

The other way of implementing (2) is to use a single multiplier for the twiddle factor multiplication as shown in Fig. 1(b). Instead of generating the vector $[F_0(k), W_N^k F_1(k), W_N^{2k} F_2(k), W_N^{3k} F_3(k)]^T$ in parallel as shown in Fig. 1(a), this scheme generates each element