Efficient FPGA and ASIC Realizations of a DA-Based Reconfigurable FIR Digital Filter

Sang Yoon Park, *Member, IEEE*, and Pramod Kumar Meher, *Senior Member, IEEE*

***Abstract*—This brief presents efficient distributed arithmetic (DA)-based approaches for high-throughput reconfigurable implementation of finite-impulse response (FIR) filters whose filter coefficients change during runtime. Conventionally, for reconfigurable DA-based implementation of FIR filter, the lookup tables (LUTs) are required to be implemented in RAM and the RAM-based LUT is found to be costly for ASIC implementation. Therefore, a shared-LUT design is proposed to realize the DA computation. Instead of using separate registers to store the possible results of partial inner products for DA processing of different bit positions, registers are shared by the DA units for bit slices of different weightage. The proposed design has nearly 68% and 58% less area-delay product and 78% and 59% less energy per sample than the DA-based systolic structure and the carry save adder (CSA)-based structure, respectively, for the ASIC implementation. A distributed-RAM-based design is also proposed for the field-programmable gate array (FPGA) implementation of the reconfigurable FIR filter, which supports up to 91 MHz input sampling frequency and offers 54% and 29% less the number of slices than the systolic structure and the CSA-based structure, respectively, when implemented in the Xilinx Virtex-5 FPGA device (XC5VSX95T-1FF1136).**

***Index Terms*—Circuit optimization, distributed arithmetic (DA), finite-impulse response (FIR) filter, reconfigurable implementation.**

I. Introduction

A RECONFIGURABLE finite-impulse response (FIR) filter whose filter coefficients dynamically change during runtime plays an important role in the software defined radio systems [1], [2], multichannel filters [3], and digital up/down converters [4]. However, the well-known multiple-constant-multiplication-based technique [5], which is widely used for the implementation of FIR filters, cannot be used when the filter coefficients dynamically change. On the other hand, a general multiplier-based structure requires a large chip area and consequently enforces a limitation on the maximum possible order of the filter that can be realized for high-throughput applications.

A distributed arithmetic (DA)-based technique [6] has gained substantial popularity in recent years for its high-throughput processing capability and increased regularity, which result in cost-effective and area-time efficient computing structures. The main operations required for DA-based computation are a sequence of lookup table (LUT) accesses followed by shift-accumulation operations of the LUT output. The conventional DA implementation used for the implementation of an FIR filter assumes that impulse response coefficients are fixed, and this behavior makes it possible to use ROM-based LUTs. The memory requirement for DA-based implementation of FIR filters, however, exponentially increases with the filter order. To eliminate the problem of such a large memory requirement, systolic decomposition techniques are suggested by Meher *et al.* for DA-based implementation of long-length convolutions and FIR filter of large orders [7], [8].

For a reconfigurable DA-based FIR filter whose filter coefficients dynamically change, we need to use rewritable RAM-based LUT [9] instead of ROM-based LUT. Another approach is to store the coefficients in the analog domain by using serial digital-to-analog converters resulting in mixed-signal architecture [10]. We also find quite a few works on DA-based implementation of adaptive filters [11], [12] where the coefficients change at every cycle. In this brief, we present efficient schemes for the optimized shared-LUT implementation of reconfigurable FIR filters using DA technique, where LUTs are shared by the DA units for bit slices of different weightage. In addition, the filter coefficients can be dynamically changed in runtime with a very small reconfiguration latency.

In the next section, we briefly discuss the mathematical background of DA-based implementation of FIR filter. In Sections III and IV, we describe the proposed reconfigurable DA-based FIR filter for ASIC and field-programmable gate array (FPGA) implementations, respectively. We provide the synthesis results of the proposed and existing designs in Section V. Finally, the conclusion is given in Section VI.

II. DA Decomposition for Implementation of an FIR Filter

The output of an FIR filter of length N can be computed as an inner product of the impulse response vector ($h(k)$, for $k = 0, 1, \ldots, N-1$) and an input vector ($x(n-k)$, for $k = 0, 1, \ldots, N-1$), which is given by

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k). \tag{1}$$

For simplification of subsequent derivation, let us remove time index n as

$$y = \sum_{k=0}^{N-1} h(k)s(k) \tag{2}$$

Manuscript received October 31, 2013; revised February 23, 2014; accepted April 24, 2014. Date of publication May 14, 2014; date of current version July 16, 2014. This brief was recommended by Associate Editor Y. Miyanaga.

S. Y. Park is with the Institute for Infocomm Research, Singapore 138632 (e-mail: sypark@i2r.a-star.edu.sg).

P. K. Meher is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: aspkmeher@ntu.edu.sg; http://www3.ntu.edu.sg/home/aspkmeher).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2014.2324418

where $s(k) = x(n-k)$. Assuming L to be the wordlength, the input sample $s(k)$ may be expressed in two's complement representation, i.e.,

$$s(k) = -[s(k)]_0 + \sum_{l=1}^{L-1} [s(k)]_l \, 2^{-l} \tag{3}$$

where $[s(k)]_l$ denotes the lth bit of $s(k)$. Substituting (3), we can write (2) in an expanded form, i.e.,

$$y = -\sum_{k=0}^{N-1} h(k) \, [s(k)]_0 + \sum_{k=0}^{N-1} h(k) \left\{ \sum_{l=1}^{L-1} [s(k)]_l \, 2^{-l} \right\}. \tag{4}$$

To convert the sum-of-products form of inner product of (2) into a distributed form, the order of summations over the indices k and l in (4) can be interchanged to have

$$y = -\sum_{k=0}^{N-1} h(k) \, [s(k)]_0 + \sum_{l=1}^{L-1} 2^{-l} \left\{ \sum_{k=0}^{N-1} h(k) \, [s(k)]_l \right\} \tag{5}$$

and the inner product given by (5) can be computed as

$$y = \sum_{l=1}^{L-1} 2^{-l} C_l - C_0 \tag{6a}$$

where

$$C_l = \sum_{k=0}^{N-1} h(k) \, [s(k)]_l \, . \tag{6b}$$

Since any element of the N-point bit sequence $[s(k)]_l$ for $0 \le k \le N-1\}$ can either be 0 or 1, the partial sum C_l for $0 \le l \le L-1$ can have 2^N possible values. If all the 2^N possible values of C_l are precomputed and stored in the LUT, the partial sums C_l can be read out from the LUT using the bit sequence $\{[s(k)]_l$ for $0 \le k \le N-1\}$ as address bits for computing the inner product.

Without a loss of generality, and for simplicity of discussion, we may assume the signal samples to be unsigned words of size L, although the proposed algorithm can be used for two's complement coding and offset binary coding also. We can always obtain unsigned input signal by adding fixed offset when the original input signal is signed. The inner product given by (6a) then can be expressed in a simpler form, i.e.,

$$y = \sum_{l=0}^{L-1} 2^{-l} C_l \tag{7}$$

so that no sign reversal of LUT output is required.

We can use (7) directly for straightforward DA-based implementation of FIR filter using the LUT containing 2^N possible values of C_l. For large values of N, however, the LUT size becomes too large, and the LUT access time also becomes large. The straightforward DA-based implementation is, therefore, not suitable for large filter orders. When N is a composite number given by $N = PM$ (P and M may be any two positive integers), one can map the index k into $(m + pM)$ for $m = 0, 1, \ldots, M-1$ and $p = 0, 1, \ldots, P-1$ to express (7) as

$$y = \sum_{l=0}^{L-1} 2^{-l} \left(\sum_{p=0}^{P-1} S_{l,p} \right) \tag{8a}$$

where $S_{l,p}$ is the sum of partial product of M samples represented as

$$S_{l,p} = \sum_{m=0}^{M-1} h(m+pM) \, [s(m+pM)]_l \tag{8b}$$

for $l = 0, 1, \ldots, L-1$ and $p = 0, 1, \ldots, P-1$.

For any given sequence of impulse response $\{h(k)\}$, the 2^M possible values of $S_{l,p}$ corresponding to the 2^M permutations of M-point bit sequence $\{(s(m+pM))_l\}$, for $m = 0, 1, \ldots, M-1$ and $l = 0, 1, \ldots, L-1$, may be stored in the LUT of 2^M words. These values of $S_{l,p}$ can be read out when the bit sequence is fed to the LUT as address. Equation (8) may, thus, be written in terms of memory-read operation as

$$y = \sum_{l=0}^{L-1} 2^{-l} \left[\sum_{p=0}^{P-1} \mathcal{F}(\mathbf{b}_{l,p}) \right] \tag{9}$$

where $\mathcal{F}(\mathbf{b}_{l,p}) = S_{l,p}$, and

$$\mathbf{b}_{l,p} = \{[s(pM)]_l \, [s(1+pM)]_l \, , \ldots, [s(M-1+pM)]_l\} \tag{10}$$

for $0 \le l \le L-1$ and $0 \le p \le P-1$. The bit vector $\mathbf{b}_{l,p}$ is used as address word for the LUT, and $\mathcal{F}(\cdot)$ is the memory-read operation.

III. Proposed Reconfigurable DA-Based FIR Filter for ASIC Implementation

The proposed structure of the DA-based FIR filter for ASIC implementation is shown in Fig. 1. The input samples $\{x(n)\}$ arriving at every sampling instant are fed to a serial-in–parallel-out shift register (SIPOSR) of size N. The SIPOSR decomposes the N recent most samples to P vectors $\mathbf{b}_p$ of length M for $p = 0, 1, \ldots, P-1$ and feeds them to P reconfigurable partial product generators (RPPGs) to calculate the partial products according to (8b). The structure of the proposed RPPG is depicted in Fig. 2 for $M = 2$. For high-throughput implementation, the RPPG generates L partial products corresponding to L bit slices in parallel using the LUT composed of a single register bank of $2^M - 1$ registers and L number of $2^M : 1$ MUXes. In the proposed structure, we reduce the storage consumption by sharing each LUT across L bit slices. The register array is preferred for this purpose rather than memory-based LUT in order to access the LUT contents simultaneously. In addition, the contents in the register-based LUT can be updated in parallel in fewer cycles than the memory-based LUT to implement desired FIR filter. The width of each register in the LUT is $(W + \lceil \log_2 M \rceil)$ bits, where W is the wordlength of the filter coefficient. The input of the MUXes are 0, $h(2p)$, $h(2p+1)$, and $h(2p) + h(2p+1)$; and the two-bit digit $\mathbf{b}_{l,p}$ is fed to MUX l for $0 \le l \le L-1$ as a control word. We can find that MUX l provides the partial product $S_{l,p}$ for $0 \le l \le L-1$ given by (8b).

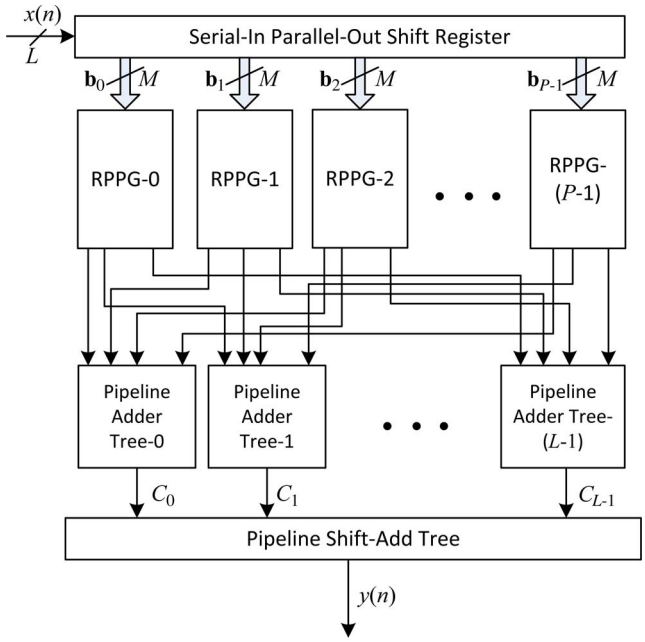

Fig. 1. Proposed structure of the high-throughput DA-based FIR filter for ASIC implementation. RPPG stands for reconfigurable partial product generator.

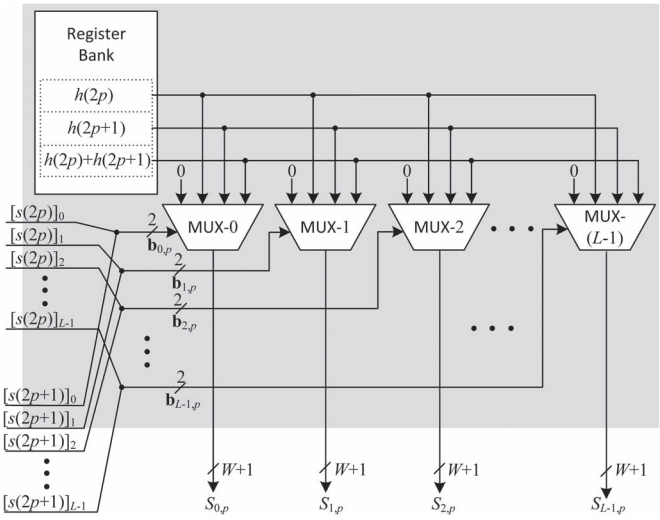

Fig. 2. pth RPPG for $M = 2$.

The $(W + 1)$-bit partial products generated by the P RPPG blocks are added by L separate pipeline adder trees (PATs) according to the inner summation in (8a). The output of PATs are appropriately shifted and added to obtain the filter output $y(n)$ by a pipeline shift-add tree (PSAT) as the outer summation in (8a). The PAT requires $P - 1$ adders in $\lceil \log_2 P \rceil$ stages and the PSAT requires $L - 1$ adders in $\lceil \log_2 L \rceil$ stages.

IV. Proposed Reconfigurable DA-Based FIR Filter for FPGA Implementation

FPGA technology has tremendously grown from a dedicated hardware to a heterogeneous system, which is considered to be a popular choice in communication base stations instead of being just a prototype platform. The proposed reconfigurable FIR filter may be also implemented as part for the complete system on FPGA. Therefore, here we propose a reconfigurable DA-based FIR filter for FPGA implementation. The architecture suggested in Section III for high-throughput implementation of DA-based FIR filter is not suitable for FPGA implementation. The structure in Fig. 1 involves $N(2^M - 1)/M$ number of registers for the implementation of LUTs for FIR filter of length N. However, registers are scarce resource in FPGA since each LUT in many FPGA devices contains only two bits of registers. Therefore, the LUTs are required to be implemented by distributed RAM (DRAM) for FPGA implementation. However, unlike the case of the RPPG in Fig. 2, the multiple number of partial inner products $S_{l,p}$ cannot be retrieved from the DRAM simultaneously since only one LUT value can be read from the DRAM per cycle. Moreover, if L is the bit width of input, the duration of the sample period of the design is L times the operating clock period, which may not be suitable for the application requiring high throughput. Using a DRAM to implement LUT for each bit slice will lead to very high resource consumption. Thus, we decompose the partial inner-product generator into Q parallel sections and each section has R time-multiplexed operations corresponding to R bit slices. When L is a composite number given by $L = RQ$ (R and Q are two positive integers), the index l in (8a) can be mapped into $(r + qR)$ for $r = 0, 1, \ldots, R - 1$ and $q = 0, 1, \ldots, Q - 1$ to modify (8a) as

$$y = \sum_{q=0}^{Q-1} 2^{-Rq} \left[\sum_{r=0}^{R-1} 2^{-r} \left(\sum_{p=0}^{P-1} S_{r+qR,p} \right) \right]. \qquad (11)$$

We have referred to the indices q and r in (11) as section index and time index, respectively. We have R time slots of the same duration as the operating clock period so that we can have one filter output every R cycles. Fig. 3(a) shows the structure of the proposed time-multiplexed DA-based FIR filter using DRAM. To implement (11), the proposed structure has Q sections, and each section consists of P DRAM-based RRPGs (DRPPGs) and the PAT to calculate the rightmost summation, followed by shift-accumulator that performs over R cycles according to the second summation. However, we can use dual-port DRAM to reduce the total size of LUTs by half since two DRPPGs from two different sections can share the single DRAM. The structure of a DRPPG is shown in Fig. 3(b). The proposed structure can produce QP partial inner products in a single cycle, whereas the structure in Fig. 1 can generate LP inner products. In the rth cycle, P DRPPGs in the qth section generate P partial inner products $S_{r+qR,p}$ for $p = 0, 1, \ldots, P - 1$ to be added by the PAT. The output of the PAT are accumulated by a shift-accumulator [see Fig. 3(c)] over R cycles. Finally, the PSAT produces the filter output using the output from each section every R cycles. The accumulated value is reset every R cycles by the control signal [acc_rst in Fig. 3(c)] to keep the accumulator register ready to be used for calculation of the next filter output. If the maximum operating clock period is f_{clk}, the proposed structure can support the input sample rate of f_{clk}/R.

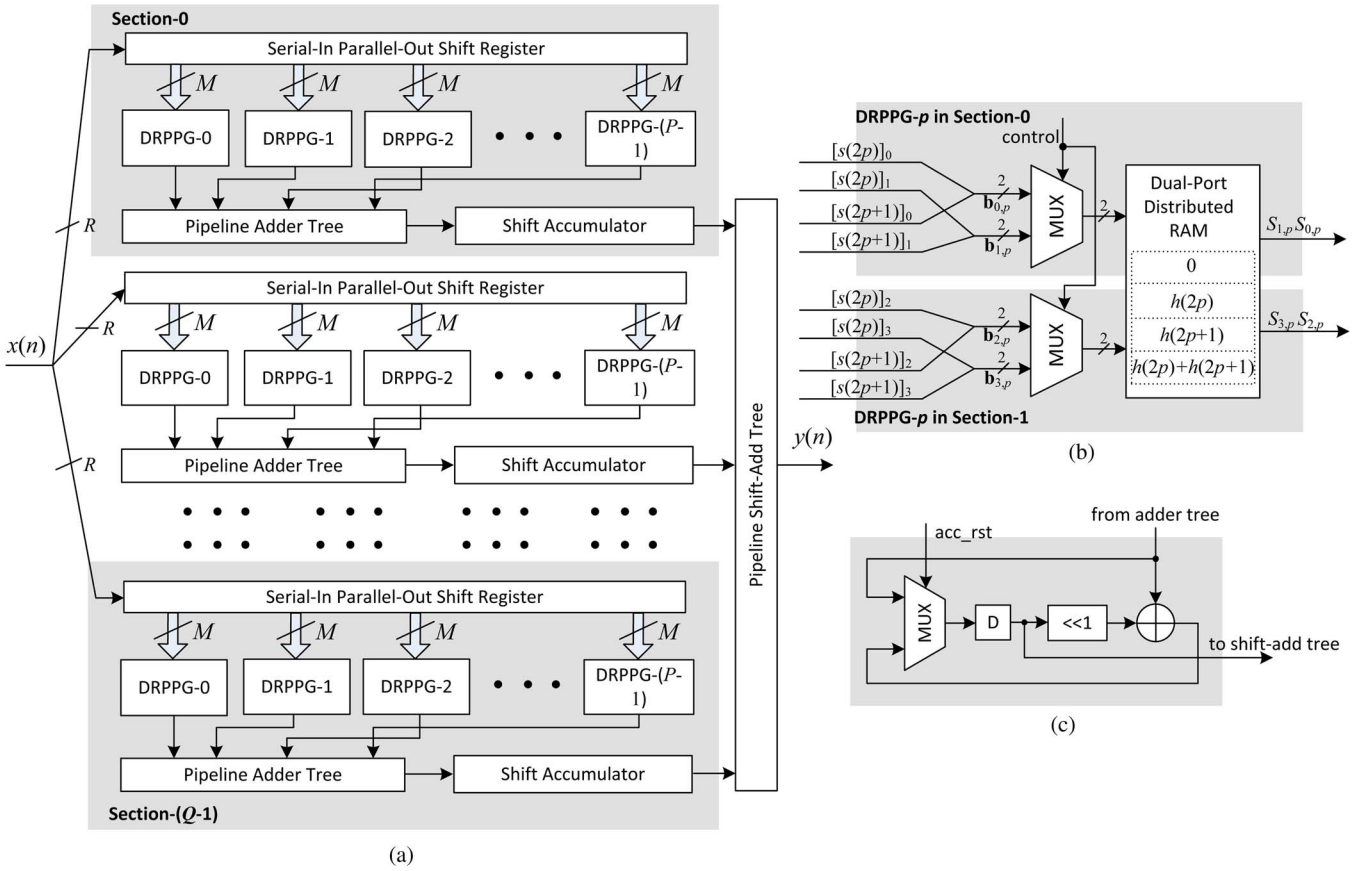

Fig. 3. Proposed structure of the DA-based FIR filter for FPGA implementation. (a) Structure of the DA-based FIR filter. (b) Structure of the DRPPG for $M = 2$ and $R = 2$. (c) Structure of the shift-accumulator.

TABLE I
HARDWARE AND TIME COMPLEXITIES OF DIFFERENT DA-BASED STRUCTURES OF AN FIR FILTER

Design	# Adders	LUT Size	# Registers	Critical Path	NOC
Structure [6]	1	2^N	1	$T_R + T_A + T_D$	$1/L$
Structure of [7], [8]	$L(P+1)$	$LP2^M$	$L(P^2+P+L+1)/2+N-1$	$T_A + T_D$	1
Structure of [12]	$3P-1$	$P(2^M-1)$	$P2^{M-1}+4P+N+1$	$MT_M + T_{FA} + T_D$	$1/L$
Structure of Fig. 1	$LP-1$	$P(2^M-1)$	$LP+N-2$	$\max\{MT_M + T'_A + T_D, T_A + T_D\}$	1
Structure of Fig. 3	$PQ+Q-1$	$PQ2^{M-1}$	$PQ+Q+N-2$	$T_A + T_D$	$1/R$

Fig. 1 and Fig. 3 are used for ASIC and FPGA implementation, respectively. NOC stands for the number of output per cycle. T_R: delay for LUT access. T_A: delay for final adder whose output wordlength is $L + W + \lceil \log_2 N \rceil$. T'_A: delay for adder with input wordlength is $L + \lceil \log_2 M \rceil$. T_D: delay for D-flip-flop. T_M: delay for 2:1 MUX. T_{FA}: delay for full adder.

V. IMPLEMENTATION RESULTS AND DISCUSSIONS

The hardware and time complexities of the proposed structures and the existing DA-based structures of an FIR filter are listed in Table I. A conventional DA-based structure [6], DA-based systolic structures [7], [8], and a DA-based structure using carry save adder (CSA) [12] are compared with the proposed structures. For the DA-based designs, the duration of minimum clock periods depends on the delay of the final adder stage in the PSAT, i.e., T_A in Table I. However, the critical path of the structure in Fig. 1 depends on the parameter M. Specifically, if M is large, the longest path might be the delay of the RPPG plus the delay of the adder in the first stage of the PAT (T'_A in Table I); otherwise, the delay of the final stage of the PSAT becomes the critical path of the filter. It is found that the proposed structure shown in Fig. 1 (using RPPG) produces the most number of output per cycle (NOC) with the systolic structure among all the listed structures. The throughput rate is the amount of output produced during a duration amounting to the critical path. Therefore, if M is small, the proposed structure in Fig. 1 offers the highest throughput, and it involves nearly L times less LUT resource than that of the systolic structure since N/M register arrays are shared to realize the LUT function corresponding to all the bit slices of different weights. The proposed structure shown in Fig. 3 (using DRPPG) for FPGA implementation guarantees a higher throughput than the structures of [6] and [12] for $R < L$ and fewer adders and smaller LUT than the systolic structure. The structure shown in Fig. 3 involves fewer adders and registers, but marginally larger LUT, than the structure shown in Fig. 1. However, for FPGA implementation, the DRAM-based LUT requires less number of slices (NOS) than the register-based LUT of the same size. It should be noted that, as M increases, the size of the LUT

TABLE II
PERFORMANCE COMPARISON OF ASIC SYNTHESIS RESULTS USING CMOS 90-nm LIBRARY FOR $L = W = 8$, $M = 2$, AND $N = 16$

Design	MSP	MSF	Area	ADP	EPS
Structure of [7]	1.79	558	71195	127439	60.76
Structure of [12]	5.20	192	18257	94936	32.38
Structure of [13]	3.32	301	51994	172620	13.63
Structure of Fig. 1	1.58	632	25163	39757	13.03

Units: MSP (ns), MSF (MHz), Area sq.um), ADP (sq.um×ns), and EPS (mW×ns)

exponentially increases, whereas the total number of adders decreases. The silicon areas of the structures in Figs. 1 and 3 are proportional to the value of P, and the filters can have more benefit due to more sharing of the LUT when L has a large value.

The proposed DA-based structure (see Fig. 1) is written in hardware description language and synthesized by Synopsys Design Compiler using the CMOS 90-nm library for ASIC implementation. The DA-based systolic structure [7], DA-based structure using CSA [12], and direct-form reconfigurable FIR filter [13] are also equivalently synthesized by Synopsys Design Compiler for comparison with the proposed design. From the synthesis result we find that the proposed design occupies the smallest silicon area when $M = 2$; therefore, the value of M is set to 2. The minimum sample period (MSP), maximum sampling frequency (MSF; reciprocal of the MSP), area, area-delay product (ADP), and energy per sample (EPS) are obtained from the synthesis result and are listed in Table II for 16-tap FIR filter $(N = 16)$, 8-bit input samples, and 8-bit filter coefficients.

The proposed structure has less area and shorter MSP compared with the DA-based systolic structure [7] and the direct-form structure of [13] and involves 68% and 76% less ADP, respectively, over the other two. The proposed structure has 1.37 times more area than the CSA-based structure [12], but 3.29 times less MSP and 58% less ADP. The proposed structure involves nearly the same (slightly less) EPS than the structure of [13], but offers nearly 78% less EPS than the structure of [7] due to much less usage of registers, as listed in Table I. The structure of [12] consumes 2.48 times more EPS than the proposed structure since it needs multiple cycles to obtain one output sample, which is an intrinsic disadvantage of the bit-serial DA-based structure.

The proposed DA-based structure in Fig. 3 for $R = 1, 2,$ and 4 and the DA-based structures of [8] and [12] are implemented on a Xilinx Virtex-5 FPGA device (XC5VSX95T-1FF1136). The Xilinx LogiCORE FIR compiler 5.0 [14] is also implemented with the architectural option of DA on the same FPGA device for comparison. The MSP, MSF, NOS, number of slice registers (SREG), number of slice LUTs (SLUT), and slice-delay product (SDP) are listed in Table III. The proposed structure for $R = 1$ involves the smallest SDP and supports the largest MSF, but requires more SLUT than the other structures. The proposed structure for $R = 2$ requires longer MSP than the structures of [8] and [14], but requires nearly 25% and 44% less NOS and involves 8% and 28% less SDP, respectively, The proposed structure for $R = 4$ requires 54%, 29%, and 65% less NOS than the structures of [8], [12], and [14], respectively, and supports up to 91 MHz MSF.

TABLE III
PERFORMANCE COMPARISON OF IMPLEMENTATION ON XILINX VIRTEX-5 (XC5VSX95T-1FF1136) FOR $L = W = 8$, $M = 4$, AND $N = 16$

Design	MSP	MSF	NOS	SREG	SLUT	SDP
Structure of [8]	4.17	239	275	688	833	1148
Structure of [12]	17.35	57	178	412	267	3088
Structure of [14]	3.96	252	368	970	806	1458
Fig. 3, $R = 1$	2.27	440	310	1120	755	804
Fig. 3, $R = 2$	5.11	195	205	671	517	1047
Fig. 3, $R = 4$	10.91	91	126	397	277	1374

VI. SUMMARY AND CONCLUSION

We have suggested efficient schemes for high-throughput reconfigurable DA-based implementation of FIR digital filters. It is shown that the hardware cost could be substantially reduced by sharing the same registers by the DA units for different bit slices. The proposed design has nearly 68% and 58% less ADP and 78% and 59% less EPS than the DA-based systolic structure and the DA-based structure using CSA, respectively, for the ASIC implementation. The proposed structure of reconfigurable FIR filter for FPGA implementation supports up to 91 MHz input sampling frequency. It is found to offer 54% and 29% less NOS than the systolic structure and the CSA-based structure, respectively.

REFERENCES

[1] T. Hentschel, M. Henker, and G. Fettweis, "The digital front-end of software radio terminals," *IEEE Pers. Commun. Mag.*, vol. 6, no. 4, pp. 40–46, Aug. 1999.
[2] K.-H. Chen and T.-D. Chiueh, "A low-power digit-based reconfigurable FIR filter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 8, pp. 617–621, Aug. 2006.
[3] L. Ming and Y. Chao, "The multiplexed structure of multi-channel FIR filter and its resources evaluation," in *Proc. Int. Conf. CDCIEM*, Mar. 2012, pp. 764–768.
[4] I. Hatai, I. Chakrabarti, and S. Banerjee, "Reconfigurable architecture of a RRC FIR interpolator for multi-standard digital up converter," in *Proc. IEEE 27th IPDPSW*, May 2013, pp. 247–251.
[5] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 9, pp. 569–577, Sep. 1995.
[6] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Mag.*, vol. 6, no. 3, pp. 4–19, Jul. 1989.
[7] P. K. Meher, "Hardware-efficient systolization of DA-based calculation of finite digital convolution," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 8, pp. 707–711, Aug. 2006.
[8] P. K. Meher, S. Chandrasekaran, and A. Amira, "FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3009–3017, Jul. 2008.
[9] M. Kumm, K. Moller, and P. Zipf, "Dynamically reconfigurable FIR filter architectures with fast reconfiguration," in *Proc. 8th Int. Workshop ReCoSoC*, Jul. 2013, pp. 1–8.
[10] E. Ozalevli, W. Huang, P. E. Hasler, and D. V. Anderson, "A reconfigurable mixed-signal VLSI implementation of distributed arithmetic used for finite-impulse response filtering," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 2, pp. 510–521, Mar. 2008.
[11] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 7, pp. 1327–1337, Jul. 2005.
[12] P. K. Meher and S. Y. Park, "High-throughput pipelined realization of adaptive FIR filter based on distributed arithmetic," in *Proc. IEEE/IFIP 19th Int. Conf. VLSI-SOC*, Oct. 2011, pp. 428–433.
[13] *DesignWare Building Block IP User Guide*, Synposys, Inc., Mountain View, CA, USA, 2012, 06-SP2.
[14] *LogiCORE IP FIR Compiler v5.0*, Xilinx, Inc., San Jose, CA, USA, 2010.