Reconfigurable FIR Filter Using Distributed Arithmetic on FPGAs

Martin Kumm, Konrad Möller and Peter Zipf
Digital Technology Group
University of Kassel, Germany
Email: {kumm, konrad.moeller, zipf}@uni-kassel.de

***Abstract*—An architecture for a dynamically run-time reconfigurable finite impulse response (FIR) filter is presented in this work. It is based on distributed arithmetic (DA) combined with a look-up table (LUT) reduction technique which allows the direct mapping to reconfigurable LUTs (CFGLUT) of the latest Xilinx FPGAs. The resulting FIR filter can be reconfigured with arbitrary coefficients which are only limited by their length and word size. The number of filter instances for reconfiguration is only limited by the block memory of the FPGA which typically allows hundreds of different configurations. The proposed reconfigurable architecture consumes 16 % less slices on average than a fixed coefficient DA filter generated by Xilinx Coregen. As the direct mapping to CFGLUTs leads to invalid filter output during reconfiguration, an alternative architecture is proposed which avoids this limitation at the cost of 19 % more slice resources on average. Using a parallel reconfiguration scheme, reconfiguration times of about 100 ns could be achieved.**

I. Introduction

Finite impulse response (FIR) filters are one of the most fundamental components in digital signal processing. Many simplifications in their hardware implementation can be made when the coefficients are constant. However, reconfigurable FIR filters for which the coefficients can be changed in run-time are required in many application scenarios like, e. g., software defined radios (SDR). This motivates the work of many researchers to extend optimization methods that were developed in the context of multiple constant multiplication (MCM) to reconfigurable multiplier blocks [1]–[8]. Such multiplier blocks are usually realized using additions, subtractions and shifts only. In a reconfigurable multiplier block, additional multiplexers are inserted in the data path to configure the multiplication with a finite set of coefficients. This saves a lot of resources as common intermediate products can be shared between different coefficient sets. However, the hardware complexity grows with the number of coefficient sets which limits the number of reconfigurable filter configurations. Typical $2 \dots 4$ coefficient sets were reported which are sufficient for many applications, e. g., for polyphase filters in SDR. However, building reconfigurable filters with many more configurations (> 10) is a very demanding task and less work was done so far in that area. Our work was motivated by the need of a reconfigurable filter that has to track the synchrotron frequency in a beam phase control system of a heavy ion synchrotron particle accelerator [9]. In that system, center frequency and band width of a band pass filter have to be adjusted hundreds of times during the acceleration cycle. The presented architecture is based on distributed arithmetic (DA) and can be reconfigured with arbitrary coefficients. Of course, it also covers smaller sets of coefficients as needed by other applications [1]–[8].

II. Run-time Reconfigurable LUTs

Xilinx provides a run-time reconfigurable 5-input LUT as primitive (CFGLUT5) for Virtex $5 \dots 7$ and Spartan 6 FPGAs [10]. It uses the same slice resources as a standard 6-input LUT but provides an additional configuration interface. The LUT can be configured as a single 5-input LUT or as two 4-input LUTs with shared inputs. The configuration interface consists of configuration data in (CDI), configuration data out (CDO), a configuration clock (CCLK) and a clock enable (CE) signal. To change the function of the LUT, CE must be set to high and a new 32 bit configuration vector must be clocked in at CDI using CCLK. Several CFGLUTs may be cascaded by connecting CDO with CDI of the next CFGLUT in a serial chain.

III. Distributed Arithmetic

The fundamental operation of a digital filter with N taps is the inner product of two vectors which can be represented as a sum-of-products of its components

$$y = \mathbf{c} \cdot \mathbf{x} = \sum_{n=0}^{N-1} c_n \, x_n \tag{1}$$

where c_n are usually constants and x_n are the time-shifted input samples. If each x_n is represented as a binary B_x bit 2'th complement number, where $x_{n,b}$ denotes the b'th bit of x_n, (1) can be rewritten to

$$y = \sum_{n=0}^{N-1} c_n \left(\sum_{b=0}^{B_x-2} 2^b x_{n,b} - 2^{B_x-1} x_{n,B_x-1} \right) \tag{2}$$

$$= \sum_{b=0}^{B_x-2} 2^b \underbrace{\sum_{n=0}^{N-1} c_n x_{n,b}}_{=f(\tilde{x}_b^N)} - 2^{B_x-1} \underbrace{\sum_{n=0}^{N-1} c_n x_{n,B_x-1}}_{=f(\tilde{x}_{B_x-1}^N)} \tag{3}$$

where $\tilde{x}_b^N = (x_{0,b}, \dots, x_{N-1,b})^T$ is a bit vector of length N containing the b'th bit of each element of $\mathbf{x}$. The function

$$f(\tilde{x}_b^N) = \sum_{n=0}^{N-1} c_n x_{n,b} \tag{4}$$

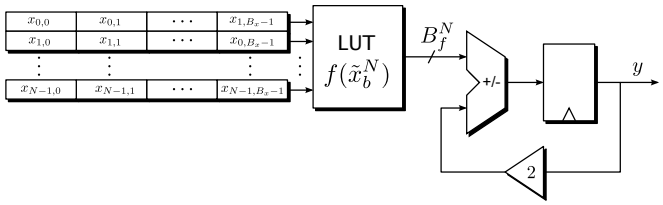

Fig. 1. Sequential realization of a distributed arithmetic FIR filter

can be precomputed and stored in a single LUT with N inputs. The storage requirement of the LUT is $B_f^N \cdot 2^N$ bit, where B_f^N denotes the output word size of the N-input LUT $f(\tilde{x}_b^N)$. The inner product can now be obtained by accumulating the shifted outputs of the LUT according to (3). A sequential realization of (3) which computes a valid output every N samples is shown in Fig. 1. For higher throughput, a parallel implementation using B_x LUTs can be obtained by unfolding.

So far, this N-input LUT can not be directly mapped to the reconfigurable 4/5-input CFGLUTs described above. Therefore, a method to reduce the LUT input size [11] was used to break the N-input LUT into several 4/5-input LUTs which is described in the following.

A. Dividing LUTs Into Smaller Partial LUTs

The input size of the LUT can be reduced by splitting the sum in (4) into several smaller sums

$$f(\tilde{x}_b^N) = \sum_{l=0}^{\lfloor N/L \rfloor - 1} \underbrace{\sum_{n=lL}^{(l+1)L-1} c_n x_{n,b}}_{f_l(\tilde{x}_b^L)} + \underbrace{\sum_{n=N-L'}^{N-1} c_n x_{n,b}}_{f_{\lfloor N/L \rfloor}(\tilde{x}_b^{L'})} \tag{5}$$

with $L < N$ where

$$f_l(\tilde{x}_b^L) = \sum_{n=lL}^{(l+1)L-1} c_n x_{n,b} \tag{6}$$

can be realized by partial L-input LUTs. If N is not dividable by L, one additional partial LUT of size $L' = N \bmod L$ is necessary, which is represented with the last term in (5). By setting $L = 4$ or $L = 5$, the LUT $f(\tilde{x}_b^N)$ can be directly mapped to CFGLUTs by using the decomposition of (5). Furthermore, this method reduces the LUT storage requirements for the N-input LUT $f(\tilde{x}_b^N)$ from $B_f^N \cdot 2^N$ bits to $\lfloor N/L \rfloor \cdot B_f^L \cdot 2^L + B_f^{L'} \cdot 2^{L'}$ bits. Note that for parallel DA, the N-input LUT is used B_x times. For a fixed L, this realization style grows linear with the number of filter taps N in contrast to (4) which grows exponentially. This memory reduction is paid by $\lfloor N/L \rfloor$ additional adders.

B. Selecting the Optimal Partial LUT Size L

As the CFGLUT5 can be configured as single 5-input LUT or two 4-input LUTs with shared inputs, the question is still open if the partial LUT size should be chosen to $L = 4$ or $L = 5$. From (5) it is clear that the inputs $\tilde{x}_b^L$ of a LUT can not be shared between several LUTs $f_l(\tilde{x}_b^L)$. But pairs of output bits of the same LUT may be realized in a single CFGLUT. To estimate the required number of CFGLUTs it is assumed in the following that N is dividable by L and B_f^L is dividable by two. Then, for each 4-input LUT with B_f^4 outputs, $B_f^4/2$ CFGLUTs are required, which leads to $N/4 \cdot B_f^4/2 = NB_f^4/8$ CFGLUTs to compute one N-input LUT $f_l(\tilde{x}_b^L)$. Using $L = 5$, for each 5-input LUT with B_f^5 outputs, B_f^5 CFGLUTs are required, which leads to $N/5 \cdot B_f^5 = NB_f^5/5$ CFGLUTs in total. The output word size of a partial LUT has to be chosen to fit the maximal possible value according to (6). This is L times $c_{n,\max} = 2^{B_c - 1} - 1$, where B_c denotes the coefficient word size, resulting in $B_f^L = \lceil \log_2(L) \rceil + B_c$. Setting $L = 4$ and $L = 5$ leads to $B_f^5 = B_f^4 + 1$ and

$$\underbrace{\frac{NB_f^5}{5} = \frac{N(B_f^4 + 1)}{5}}_{\text{CFGLUTs of } f_l(\tilde{x}_b^5)} > \underbrace{\frac{NB_f^4}{8}}_{\text{CFGLUTs of } f_l(\tilde{x}_b^4)} \tag{7}$$

which means that $L = 4$ always leads to less resources with the assumptions above, so this was used for the proposed architecture. If N is not dividable by L, $B_f^{L'}$ extra CFGLUTs are required in general. If the word size B_f^L is not dividable by two, one half of a CFGLUT in case $L = 4$ is unused. For large N and large B_L these terms can be neglected.

C. Coefficient Symmetry

If the FIR filter has a linear phase, which is usually the case for common filter design methods, the number of CFGLUTs in parallel DA can be further reduced by exploiting the symmetry in the coefficients which has the form

$$c_n = \pm c_{N-n-1} \ . \tag{8}$$

This can be used to nearly halve the number of LUT inputs. For even N, (1) can be rewritten to

$$y = \sum_{n=0}^{N/2-1} c_n \left(x_n \pm x_{N-n-1} \right) \ , \tag{9}$$

and for odd N, (1) results in

$$y = \sum_{n=0}^{(N-1)/2-1} c_n \underbrace{\left(x_n \pm x_{N-n-1} \right)}_{=z_n} + c_{(N-1)/2} \underbrace{x_{(N-1)/2}}_{=z_{M-1}} \tag{10}$$

The N sum terms of (1) are reduced to $M = \lceil \frac{N}{2} \rceil$ terms in (9) and (10). This approximately halves the input size of LUT $f(\tilde{x}_b^N)$ which halves the number of CFGLUTs while M additional adders are needed.

IV. Reconfigurable DA Architectures

A. Resource Optimized Architecture

The proposed reconfigurable parallel DA filter which uses the optimization methods of the last section is shown in Fig. 2. Note that all adders are followed by pipeline registers and consecutive adders are realized as a pipelined adder tree. Many shift operations of the output adder tree can be moved towards the output for word size reduction (not shown in Fig. 2). The reconfigurable LUT (RLUT) is shown in Fig. 3. It consists

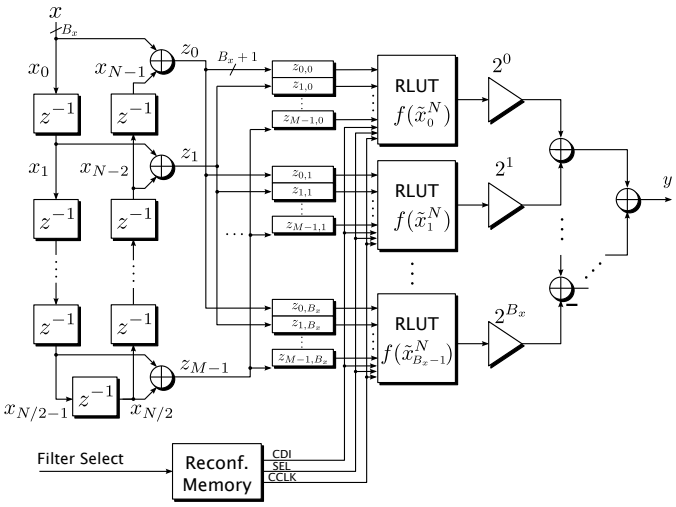

Fig. 2. Architecture of the reconfigurable distributed arithmetic FIR filter

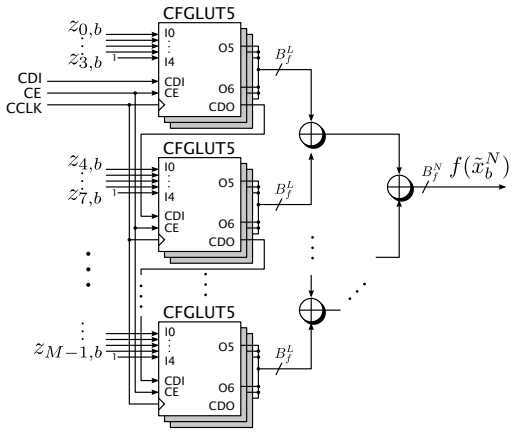

Fig. 3. A reconfigurable LUT realization of $f(\tilde{x}_b^N)$ using single CFGLUT5 primitives for the resource minimized DA architecture

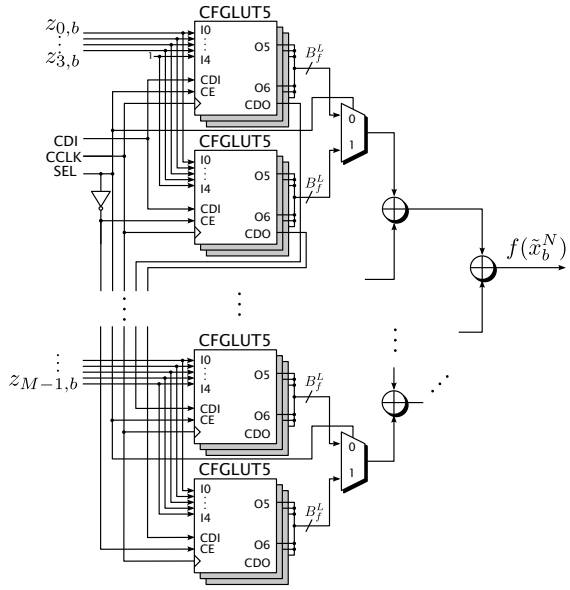

Fig. 4. A reconfigurable realization of $f(\tilde{x}_b^N)$ using twice as many CFGLUT5 primitives for the glitch free reconfiguration architecture

of CFGLUT5 primitives which are configured as two 4-input LUTs, followed by a pipeline register (not shown). Hence, each CFGLUT5 computes two bits of $f(\tilde{x}_b^L)$ which are further processed in a pipelined adder tree according to (5). In Fig. 3, all CFGLUTs are cascaded using a serial chain with CDI and CDO. Alternatively, they can be configured in parallel such that each CFGLUT5 has its own CDI wire. These two cascading schemes are called serial or parallel configuration schemes in the following. While the serial configuration scheme requires only local wires, many long wires are needed in the parallel configuration scheme. However, in the parallel configuration scheme, the configuration time is 32 clock cycles while in serial scheme, this is multiplied by the number of CFGLUTs which are included in one RLUT. As all RLUTs in Fig. 2 have the identical content, the configuration interface can be connected in parallel. This greatly reduces configuration memory and configuration time. A reconfigurable sequential DA realization can be obtained by simply replacing the LUT in Fig. 1 with a reconfigurable LUT.

In the proposed architecture, any number of taps up to N with coefficients up to the word size B_c can be configured. Lower N and B_c can be achieved by simply setting the corresponding bit positions in the coefficients to zero. Note that all configurations must have the same symmetry in their coefficients. This can be easily achieved in the initial filter design.

B. Glitch Free Reconfiguration Architecture

The RLUT implementation of Fig. 3 has one drawback: the output of each CFGLUT is invalid during reconfiguration. This may be a significant limitation for many applications. Thus, an alternative RLUT implementation which uses two banks of CFGLUTs is proposed, which is shown in Fig. 4. Each CFGLUT is replaced by two CFGLUTs. As shown later, this does by far not double the total resources. The clock enable of one CFGLUT is connected with the bank select (SEL) signal, the other one with its inverse $\overline{\text{SEL}}$. The same signal controls a multiplexer that selects the output of the CFGLUT which is currently *not* reconfigured. With these simple modifications, a secure reconfiguration can be performed by first shifting the new configuration into the RLUT and second, toggling the bank select signal. With this mechanism, the next configuration can be prepared and then activated in a single clock cycle.

V. Results

To evaluate the resource usage and speed of the proposed reconfigurable parallel DA architectures, several synthesis experiments were performed. For that, a VHDL code generator was written. All synthesis results were obtained for the smallest Xilinx Virtex 6 FPGA (XC6VLX75T-2FF484-2) after place & route using Xilinx ISE v13.4. To ease the comparison with fixed coefficient filters, we used a benchmark set of nine filters which were already used in previous publications [12]–[14]. The coefficient bit width is $B_c = 17$ bit and the input bit width was chosen to $B_x = 12$ bit, like in the previous

TABLE I
SYNTHESIS RESULTS FOR THE PROPOSED RESOURCE OPTIMIZED AND GLITCH FREE (GF) RECONFIGURABLE DA ARCHITECTURES AS WELL AS THE FIXED COEFFICIENT DA REALIZATION OF COREGEN

		Reconf. DA			GF Reconf. DA			Coregen DA	
N	S [bit]	Slices	f_{clk} [MHz]	T_{cclk} [ns]	Slices	f_{clk} [MHz]	T_{cclk} [ns]	Slices	f_{clk} [MHz]
6	320	195	480.1	2.08	224	506.8	1.97	182	499
10	608	265	461.5	2.17	381	407.7	2.45	271	474
13	640	292	466.9	2.14	342	514.9	1.94	302	429
20	928	427	449.0	2.23	549	474.4	2.11	453	456
28	1248	603	420.3	2.38	692	468.4	2.13	655	413
41	1888	914	389.9	2.56	1189	450.5	2.22	1004	457
61	2560	1188	370.9	2.7	1341	395.6	2.53	1391	411
119	4800	2155	306.0	3.27	2517	308.1	3.25	2693	352
151	6080	2776	301.5	3.32	3268	353.7	2.83	3574	306
avg.:	2119.1	979.4	405.1	2.5	1167	431.1	2.38	1169.4	421.9

work. The same benchmark was used for generating parallel DA filters using the FIR Compiler v5.0 tool [15] of Xilinx Coregen. These were used to analyze the overhead of the proposed reconfigurable DA architectures compared to a fixed coefficient DA. This overhead is expected due to the following reasons: 1) the word size of $f(\tilde{x}_b^L)$ can be reduced by many bits when the coefficients c_n are known in advance which directly reduces the number of partial LUTs, 2) the resulting fixed LUTs can be further reduced by logic optimization [16].

The synthesis results including slice resources, maximum clock frequency (f_{clk}), the minimal period of the configuration clock (T_{cclk}) as well as the storage requirements per filter (S) are listed in Table I. As the filter clock and the reconfiguration clock resulted in a similar timing performance, they were joined to a single clock. This enables to use the flip-flop inside the same slice of the CFGLUT for pipelining. Otherwise, another slice in a different clock region is instantiated. The reconfiguration circuit is not included in Table I as it depends on the number of filters and their access scheme (e. g., successive or random). But even for the largest filter in the benchmark ($N = 151$) using 100 configurations, a parallel reconfiguration circuit consisting of a simple finite state machine for random filter addressing took 124 slices plus 684 kbit of block ram (12% of the smallest Virtex 6). The reconfiguration time is either $S \cdot T_{\text{cclk}}$ using the serial reconfiguration scheme or $32 \cdot T_{\text{cclk}}$ using the parallel reconfiguration scheme. Thus, the reconfiguration times are in the range of 0.7 μs to 20.2 μs and 67 ns to 106 ns for the serial and parallel reconfiguration scheme, respectively.

Surprisingly, the resource optimized reconfigurable DA needs 16% less resources at a comparable speed than the static DA produced by Coregen. Although the CFGLUTs are doubled in the glitch free architecture it consumes only 19% more slice resources on average than the resource optimized architecture. Its resource and frequency results are still in the same order of the Coregen designs.

VI. CONCLUSION

A reconfigurable FIR filter based on distributed arithmetic was presented which can be reconfigured with an arbitrary large number of filters which is only limited by the configuration memory. Only the worst parameters of filter length N, coefficient word size B_c and input word size B_x have to be known at design time. Two alternative architectures were analyzed, one resource optimized architecture, where the filter can not be used during reconfiguration and one glitch free architecture without that limitation. It was shown that the penalty for a glitch free reconfiguration is 19% on average compared to the resource optimized reconfigurable DA. Comparisons with Xilinx Coregen have shown that there is much optimization potential for resource reductions in their tool for static coefficients (at least for Virtex 6). Compared to the standard internal configuration access port (ICAP) of Xilinx [17], the reconfiguration time is greatly reduced to 32 clock cycles using a parallel reconfiguration scheme and reconfiguration memory is reduced due to the intrinsic duplicated LUT content.

REFERENCES

[1] S. S. Demirsoy, A. Dempster, and I. Kale, "Design guidelines for reconfigurable multiplier blocks," in *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, 2003.

[2] S. S. Demirsoy, I. Kale, and A. Dempster, "Efficient implementation of digital filters using novel reconfigurable multiplier blocks," in *Signals, Systems and Computers, 2004. Asilomar Conference on*, 2004.

[3] ——, "Synthesis of reconfigurable multiplier blocks: part I - fundamentals," *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pp. 536–539, 2005.

[4] ——, "Synthesis of reconfigurable multiplier blocks: part - II algorithm," *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pp. 540–543, 2005.

[5] P. Tummeltshammer, J. Hoe, and M. Puschel, "Time-Multiplexed Multiple-Constant Multiplication," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 9, pp. 1551–1563, Sep. 2007.

[6] J. Chen, C.-H. Chang, and C.-C. Jong, "Time-multiplexed data flow graph for the design of configurable multiplier block," *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pp. 1145–1148, 2009.

[7] J. Chen and C.-H. Chang, "High-Level Synthesis Algorithm for the Design of Reconfigurable Constant Multiplier," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 12, pp. 1844–1856, Dec. 2009.

[8] M. Faust, O. Gustafsson, and C.-H. Chang, "Reconfigurable multiple constant multiplication using minimum adder depth," in *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on*, 2010, pp. 1297–1301.

[9] H. Klingbeil, B. Zipfel, M. Kumm, and P. Moritz, "A digital beam-phase control system for heavy-ion synchrotrons," *Nuclear Science, IEEE Transactions on*, vol. 54, no. 6, pp. 2604–2610, 2007.

[10] Xilinx, Inc., *Xilinx Virtex-5 Libraries Guide for HDL Designs*, Aug. 2009.

[11] White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," *ASSP Magazine, IEEE*, vol. 6, no. 3, 1989.

[12] S. Mirzaei, R. Kastner, and A. Hosangadi, "Layout aware optimization of high speed fixed coefficient FIR filters for FPGAs," *Int. Journal of Reconfigurable Computing*, vol. 3, pp. 1–17, Jan 2010.

[13] M. Kumm and P. Zipf, "High Speed Low Complexity FPGA-based FIR Filters Using Pipelined Adder Graphs," in *Field Programmable Technology, Int. Conf. on (ICFPT)*, 2011.

[14] U. Meyer-Baese, G. Botella, D. Romeroa, and M. Kumm, "Optimization of High Speed Pipelining in FPGA-based FIR Filter Design using Genetic Algorithm," in *SPIE Defense Security+Sensing*, 2012.

[15] Xilinx Inc., *IP LogiCORE FIR Compiler v5.0, DS534*, 2011.

[16] M. Kumm, K. Möller, and P. Zipf, "Partial LUT Size Analysis in Distributed Arithmetic FIR Filters on FPGAs," in *Circuits and Systems, IEEE Int. Sym. on (ISCAS)*, 2013.

[17] Xilinx, Inc., *Partial Reconfiguration User Guide, UG702*, October 2010.