LOW POWER IMPLEMENTATION OF DIGITAL PREDISTORTION FILTER ON A HETEROGENEOUS APPLICATION SPECIFIC MULTIPROCESSOR

Amanullah Ghazi[1] , Jani Boutellier[1], Mahmoud Abdelaziz[2], Xiaojia Lu[1], Lauri Anttila[2], Joseph R. Cavallaro[3], Shuvra S. Bhattacharyya[4], Mikko Valkama[2], Markku Juntti[1]

[1]University of Oulu, Dept. Computer Science and Engineering, Finland
[2]Tampere University of Technology, Dept. Electronics and Communications Engineering, Finland
[3]Rice University, ECE Department, Houston, TX
[4]University of Maryland, ECE Department, College Park, MD

ABSTRACT

Power-constrained mobile radio communication transmitters drive their transmit power amplifiers close to their saturation regions, which results in nonlinear intermodulation distortion that is especially harmful in multi-cluster and carrier aggregation transmission scenarios. Digital predistortion is a method for linearizing the transmitter and suppressing the most harmful spurious emissions at the transmitter power amplifier output. This paper describes a programmable implementation of a digital predistortion filter on a heterogeneous Transport Trigger Architecture (TTA) multiprocessor. The predistortion algorithm is based on a parallel Hammerstein polynomial model and the experimental results show that the proposed programmable architecture is capable of linearizing a 20 MHz LTE carrier in real-time with a power consumption that is suitable for mobile devices.

Index Terms— Predistortion, Digital signal processing, Multicore processing

1. INTRODUCTION

Mobile radio communication transmitters often use complex in-phase and quadrature phase mixers [1] that provide the flexibility needed for building software defined radio (SDR) systems. However, these direct conversion transceivers have several imperfections arising from the nonlinearities and non-ideal behavior of analog RF and digital baseband transceiver components. Moreover, power-constrained mobile transmitters drive their transmit power amplifiers close to their saturation regions [2], which results in nonlinear intermodulation distortion that is especially harmful in multi-cluster and carrier aggregation (CA) transmission scenarios [3].

Digital predistortion (DPD) can be used to compensate these impairments and to suppress the most harmful spurious emissions at the transmitter power amplifier output by predistorting the baseband signal before transmission.

Building flexible radio transceivers requires the use of configurable (ideally software programmable) components. Traditionally, signal processing algorithms requiring high throughput have been implemented as hardwired circuits. Hard-wired implementations consume minimum energy but are slow to design and lack the flexibility of programmable implementations. On the other hand, off-the-shelf Digital Signal Processors provide programmability, but unfortunately have higher cost and energy consumption compared to hardwired implementations.

Application specific processors offer a compromise between these two extremes. They provide a low-cost and low-energy platform for programmable implementation of signal processing algorithms. This paper describes an application-specific multiprocessor for digital predistortion. The processors are based on the Transport Triggered Architecture (TTA) paradigm [4, 5], which provides programmability with high energy-efficiency [6].

The results of this paper show that digital predistortion can be implemented by programmable hardware with a power consumption that is suitable for mobile devices. Interestingly, the resulting multiprocessor architecture closely resembles that of mobile graphics processors [7], which could open up a possibility for sharing computational resources on mobile devices that often are power- and cost-limited.

2. DIGITAL PREDISTORTION

The proposed predistortion filter design is based on the algorithm presented by Anttila et al. [8], but has been modified slightly compared to the original version in order to save on the number of computations required. The initial feasibility study of the DPD filter has been performed by considering different DPD parameters and estimating processing resources [9]. Based on the feasibility study, a feasible use case of the DPD filter has been selected and implemented. This paper presents the design and results of the implemented DPD filter.

The structure of the predistortion algorithm is presented in Fig. 1. The predistortion filter is based on a parallel Hammerstein (PH) model with polynomial nonlinearities, given as [8]

$$f(x_n) = \sum_{p \in I_P} f_{p,n} * \psi_p(x_n) \qquad (1)$$

where x_n is the input sample, $f_{p,n}$ are FIR filters with M taps each, * denotes convolution, and I_P is the set of polynomial orders used.

The *polynomial basis functions* are defined as

$$\psi_p(x_n) = \sum_{k \in I_P} u_{k,p} \, |x_n|^{k-1} x_n, \quad p \in I_P, \qquad (2)$$

This work was supported in part by the US National Science Foundation under grants CNS–1265332 and CNS–1264486 as well as by the Finnish Funding Agency for Technology and Innovation under the project "Cross-Layer Modeling and Design of Energy-Aware Cognitive Radio Networks (CREAM)".

with $u_{k,p}$ denoting the polynomial weights. If all polynomial orders up to P are considered, $I_P = \{1, 2, 3, \ldots, P\}$, and when only odd order polynomials are used, $I_P = \{1, 3, 5, \ldots, P\}$. The square root

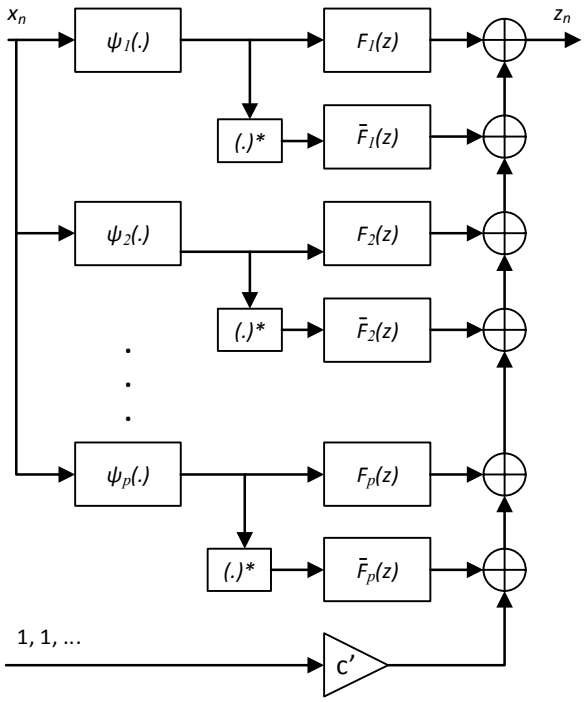

Fig. 1. Predistorter structure.

computation while computing $|x_n|^{k-1}$ can be avoided if only odd-order polynomials are used, which is a computation-saving option that has been chosen in the proposed implementation.

The polynomial weights $u_{k,p}$ determine the type of the polynomials used. The basis function description in (2) allows using either conventional polynomials (when setting $u_{k,p} = 0$, $k \neq p$), or orthogonal polynomials, such as those described in [10].

The maximum polynomial order used can be different for the conjugate and non-conjugate branch of the predistorter. When using two sets of polynomials, for the non-conjugate branch $I_P = \{1, 3, 5, \ldots, P\}$ and for the conjugate branch $I_Q = \{1, 3, 5, \ldots, Q\}$, the output of the predistortion filter can be given as

$$z_n = \sum_{p \in I_P} f_{p,n} * \psi_p(x_n) + \sum_{q \in I_Q} f_{q,n} * \overline{\psi}_q(x_n) + c', \qquad (3)$$

where $f_{p,n}$ are the filter coefficients and c' is the estimated compensation for LO leakage. [8]

The filters $F_P(z)$, $F_Q(z)$ are FIR filters whose coefficients are estimated considering the impairments from the power amplifier and the IQ modulator. The number of filters in the non-conjugate branch (N_P) and in the conjugate branch (N_Q) can be the same or differ. [8]

3. PREDISTORTION FILTER DESIGN

As a motivational example for predistortion we consider a predistortion filter with polynomial order 5 and maximum filter lengths of 5. The sampling rate is set to 92.16 MHz, which enables linearizing either a) one 20 MHz LTE carrier, b) two 10 MHz LTE carriers through contiguous CA, or c) four 5 MHz LTE carriers through contiguous CA. In this section we present a programmable predistorter implementation that adheres to these requirements.

3.1. Dataflow modeling of predistortion filter software

Initially, the predistortion filter was modeled using a dataflow description based on a specific, DSP-oriented dataflow programming model called *lightweight dataflow* [11] later to be implemented on separate processors. Following the dataflow paradigm, node computation is managed using first-in-first-out (FIFO) buffers.

The synchronous dataflow model [12] (that is supported by lightweight dataflow) is well-suited for modeling the proposed predistortion filter, as the nodes of the dataflow graph (see Fig. 2) have constant sample rates. The dataflow implementation of the algorithm consists of two dataflow node types, namely 1) computation of polynomial basis functions ψ_P, ψ_Q and 2) FIR filtering $F_P(z)$, $F_Q(z)$.

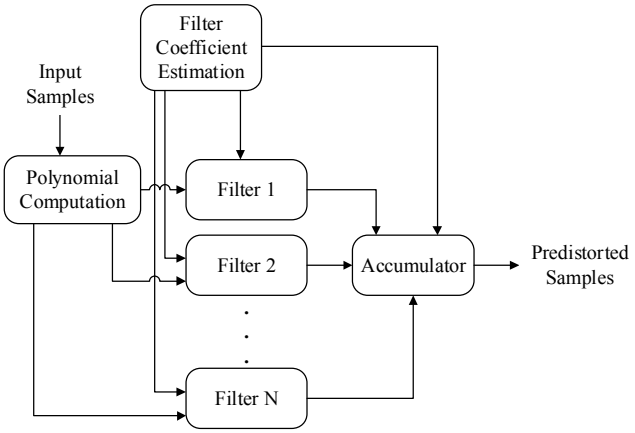

Fig. 2. Dataflow model of the predistortion filter.

3.1.1. Polynomial computation

The dataflow node for computing polynomials reads samples coming from the transmitter baseband through an input FIFO and computes the polynomial basis functions using equations (1) and (2). The polynomial weights ($u_{k,p}$) are assumed to be precomputed, and since only odd polynomial orders are used, computation of $|x_n|^{k-1}$ does not require square root computations.

The dataflow node responsible for computing the polynomials also distributes the computed polynomial values to the filters $F_P(z)$, $F_Q(z)$. The node also conjugates the computed polynomial values for $F_Q(z)$ before writing to the output FIFO.

3.1.2. FIR filtering

The traditional *shift-register* based FIR filter implementation needs shifting of the input values during each iteration which complicates the parallelization of the filter. To enable parallel filter implementation, the FIR filter is implemented using *ping-pong* buffers and a *windowing* technique.

Initially, samples are read into the p buffer while the p' buffer contains the previous samples (zeros during the first iteration). A *filtering window* is defined containing the current and previous samples needed for FIR filtering. Once the filtering window reaches the end of the p buffer, the next input sample is read to the beginning of the p' buffer. p' and p buffers are then interchanged

for the next iteration. An iteration of *ping-pong* buffer based FIR filtering is shown in Fig. 3.

With *ping-pong* buffering of the FIR filter, shifting of input samples from one register to another is avoided. Also, the software compiler has better chances of parallelizing the algorithm as multiple input samples are allowed to be read at once and more than one filtering windows is in use concurrently. Note that the buffers are allocated to the processors' internal register files and therefore the processors for FIR filtering do not need data memory.

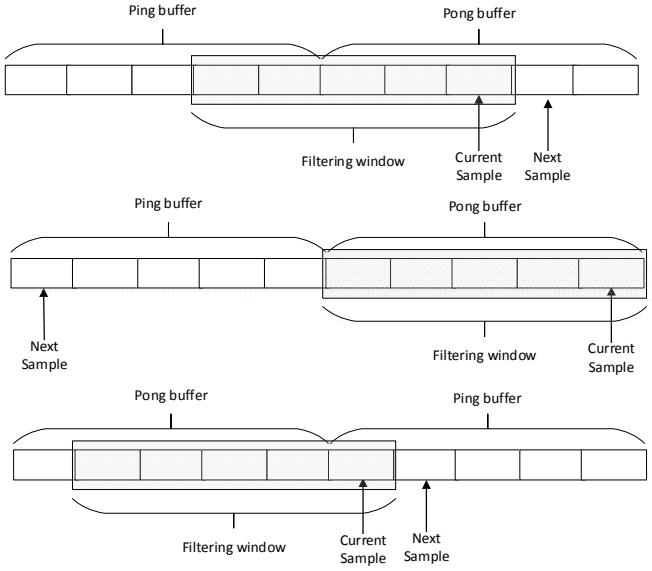

Fig. 3. FIR filtering with ping-pong buffers.

3.1.3. Accumulator

The accumulator reads filtered samples from each filter and sums them together. The accumulator also acquires the LO leakage compensation value (*c'*) from the filter parameter estimator (that is not a part of our design) and adds it to the final output. The accumulator can be implemented in non-programmable hardware for the obvious reason of its simplicity.

3.1.4. Parameter Estimation

The predistorter parameters (filter coefficients, LO leakage) need to be estimated and provided to the predistortion filter. The parameters need to be updated when the operating conditions (e.g. temperature, PA operating point, carrier frequency) of the transmitter change. This can be done in parallel with the operation of the predistortion filter and is not considered to be on the critical path of the digital predistortion system.

In the presented design the predistorter parameters are read to the FIR filters from a specific input FIFO at filter initialization. Thus, run-time update of the parameters was not considered in the experiments presented here.

3.2. TTA processor architecture

The programmable processors used in our predistorter design are of the Transport Triggered Architecture type. TTA processors resemble Very Long Instruction Word processors (VLIW) in the sense that they fetch and execute multiple instructions each clock cycle. Moreover, TTA processors provide an *exposed datapath* where the compiler directly programs the data transports inside the processor. Direct programming of data transports reduces register file traffic when compared to conventional VLIWs [4] and simplifies the processor architecture as instruction scheduling and register assignment decisions are made off-line by the compiler. Hence there are savings on gate count and energy consumption.

Design of application-specific TTA processors for the proposed design was done by the open-source TTA Codesign Environment (TCE) toolset [5]. The TCE toolset enables the designer to fully define the number of function units, register files and processor buses. Furthermore, the processor can also be equipped with custom instructions and use an arbitrary data word size. In the presented design, no custom instructions are used; all processor instructions are generic and can be used for running other applications as well.

3.3. Half-precision floating point arithmetic

Fixed-point computations are not well-suited for predistortion filtering due to the polynomial basis function computations that require computing of exponentials and thus a high dynamic range. For this reason, a floating point data type is used throughout the proposed implementation. Unfortunately, the single (32-bit) and double precision (64-bit) floating point operations have a high computational latency (5 clock cycles for 32-bit in our processor design framework) even when implemented in hardware.

Fortunately, satisfactory computational latency and dynamic range is achieved by using the *half-precision* floating point number format. The IEEE 754 standard defines 16-bit half-precision floating numbers that have 5 bits for the exponent, 10 bits for the fraction and 1 bit for the sign [13]. Previously, the half-precision floating point format has been used on a TTA processor by Janhunen et al. [6]. The added noise from using the 16-bit floating point format is discussed in Section 4.

3.4. TTA multiprocessor design

The computational complexity of the predistortion filter depends on multiple parameters. The maximum orders of the polynomials for non-conjugate (P) and conjugate (Q) branches determine the *number of filters* needed. Since we are using only odd-order polynomials, the number of filters is given by

$$N_{flt} = \frac{P+1}{2} + \frac{Q+1}{2} \tag{4}$$

Computation of the polynomial basis functions is a sequential procedure and thus performed on a single processor. As the maximum order of polynomials is much smaller than the number of FIR filter taps, the computation of polynomials is less complex than FIR filtering.

Two types of TTA processors were needed; one processor type is specifically designed for polynomial computation and the processor type is specifically for FIR filtering. Heeding the dataflow paradigm, each processor executes its program independently and communicates with other processors over FIFO buffers only. Resulting multiprocessor system is similar to the design presented in [14].

Table 1. Predistortion filter specification.

Parameter	P branch	Q branch
Max. polynomial order	5	3
Number of filters	3	2
Taps per filter	5	5

Table 2. Processing resources for TTA processors.

Processing resources	Polynomial comp. processor	FIR filter processor
Multipliers	8	12
Add/subtractors	3	16
Integer ALU	1	1
FIFO I/O units	12	5
Reg. Files (8 slots each)	5	15
Buses	18	33

4. IMPLEMENTATION AND RESULTS

A predistortion filter (with the parameters listed in Table 1) was implemented as a TTA multiprocessor. One processor was needed for polynomial computations and five processors for FIR filtering. The processors are software programmable and can thus be reconfigured or be used for other computations as well.

Having a separate processor for each FIR filter makes the design scalable: the polynomial order of the implemented predistorter can be increased by adding more FIR processors as far as the performance of the polynomial computation processor suffices.

The processing resources used in both processor types are listed in Table 2. All the resources have a word length of 16 bits; the multipliers and add/sub units perform half-precision 16-bit floating point operations except for the Arithmetic-Logical Unit (ALU) that supports mandatory integer operations.

Table 3. Synthesis results for 90nm low-leakage CMOS.

	Polynomial comp.	FIR
f_{MAX} (MHz)	125 MHz	125 MHz
Gates (NAND eq.)	71247	165214
Average Power (mW)	11.3	26.8
Energy / sample (nJ)	0.25	0.69
Throughput Ms/s	44.6	39.1

Table 4. Synthesis results for 45nm CMOS.

	Polynomial comp.	FIR
f (MHz)	320 MHz	320 MHz
Gates (NAND eq.)	62404	116822
Throughput Ms/s	114.3	100.0

4.1. Processor synthesis results

The two predistortion filter processor types were synthesized separately using the UMC 90 nm low-leakage low-k standard cell library [15] (FO4 delay estimated as 41 ps) and the Synopsys Design compiler. The operating conditions (temperature, operating voltage, manufacturing process quality) for synthesis were set to default values (TCCOM). The synthesized processors were simulated using Mentor Graphics ModelSim and the power consumption measurements were performed using Synopsys PrimeTime with signal transitions acquired from simulation. The synthesis results for the two processors are listed in Table 3. The total power dissipation for the six processors was 145.3 mW without memories.

The FIR filter processor takes 3.2 cycles/sample to execute and the polynomial computation processor is somewhat faster: 2.8 cycles/sample. In the motivational example (Section 3) a 92.16 Msamples/s input signal is predistorted with a filter of the specifications given in Table 1. However, Table 3 shows that the implementation remains from the required performance by a factor of 2.3. Fortunately, this gap in computational performance can be compensated by 1) moving from the low-leakage standard cell library to a standard performance library [16] (for UMC 90 nm FO4 = 41 ps → FO4 = 25 ps [17]) and 2) using a more up-to-date technology node such as 45 nm (FO4 = 16.7 ps [18]). To verify that the throughput requirements can indeed be met with 45nm technology, the two processors are synthesized with TSMC 45nm standard cell library with a target frequency of 320 MHz. The 45nm synthesis results are presented in Table 4.

4.2. Linearization performance

The proposed design uses a reduced-accuracy 16-bit floating point number format, whose effect on the algorithm performance needs to be explicitly analyzed.

As a reference, a random white Gaussian noise input signal was filtered with the same algorithm implemented using 64-bit double precision floating point format. The reference output was then compared against the 16-bit floating point result of the same random signal and the signal-to-noise ratio (SNR) compared to the reference computation was measured to be 46.5 dB.

Considering this limited computational accuracy (SNR) of the DPD, a simulation was performed for a CA scenario of two 10 MHz LTE carriers with 5 Resource Blocks allocated per carrier (i.e. 1 MHz BW per carrier). Fig. 3. shows the resulting overall intermodulation suppression and Fig. 4. a more detailed view with the LTE spurious emission limit that is achieved with the proposed filter. In the simulation, the power amplifier 1 dB compression point was assumed to be 26.5 dBm and modeled with a 5th order Wiener architecture. The transmission power was set to 22.0 dBm.

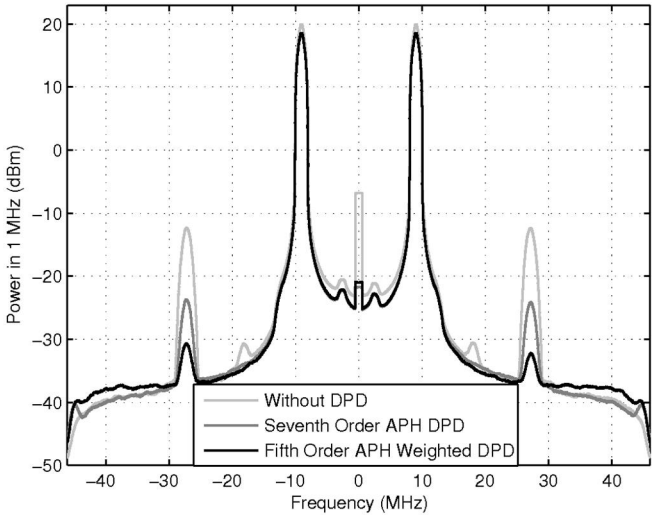

Fig. 4. Intermodulation suppression without predistortion, with classical 7th order wideband linearization and the predistorter used in the presented design (weighted DPD).

5. CONCLUSION

We have presented a programmable heterogeneous multiprocessor for digital predistortion filtering. The processor is capable of linearizing a 20 MHz LTE carrier in real time with a power dissipation that makes the solution feasible for mobile devices.

6. REFERENCES

[1] P.-I. Mak, S.-P. U, R. P. Martins, "Transceiver architecture selection: review, state-of-the-art survey and case study," *IEEE Circuits and Systems Magazine*. IEEE, 2007, vol.7, pp. 6-25.

[2] A. Katz, R. Gray, and R. Dorval, "Truly wideband linearization," *IEEE Microwave Magazine*, IEEE, 2009, vol. 10, pp. 20-27.

[3] E. Dahlman, S. Parkvall, J. Sköld, "4G LTE/LTE-Advanced for Mobile Broadband," Elsevier Ltd., 2011.

[4] H. Corporaal, "Microprocessor architectures: from VLIW to TTA," J. Wiley, 1998.

[5] O. Esko, P. Jääskeläinen, P. Huerta, C. S. de la Lama, J. Takala, J. I. Martinez, "Customized exposed datapath soft-core design flow with compiler support", *Proc. International Conference on Field Programmable Logic and Applications*. IEEE, 2010, pp. 217-222.

[6] J. Janhunen, T. Pitkänen, O. Silvén, M. Juntti, "Fixed- and floating-point processor comparison for MIMO-OFDM detector," *IEEE Journal of Selected Topics in Signal Processing*, IEEE, 2011, vol. 5, pp. 1588-1598.

[7] Arm Ltd. "Mali-400 MP: A Scalable GPU for Mobile Devices," http: //www.highperformancegraphics.org/previous/ www_2010/media/Hot3D/HPG2010_Hot3D_ARM.pdf.

[8] L. Anttila, P. Händel, M. Valkama, "Joint mitigation of power amplifier and I/Q modulator impairments in broadband direct-conversion transmitters," *IEEE Transactions on Microwave Theory and Techniques*. IEEE, 2010, vol.58, pp. 730-739.

[9] M. Abdelaziz, A. Ghazi, L. Anttila, J. Boutellier, T. Lähteensuo, X. Lu, J.R. Cavallaro, S.S. Bhattacharyya, M. Juntti, M. Valkama, "Mobile transmitter digital predistortion: feasibility analysis, algorithms and design exploration", Asilomar Conference on Signals, Systems, and Computers, 2013.

[10] R. Raich, G. T. Zhou, "Orthogonal polynomials for complex Gaussian processes," *IEEE Transactions on Signal Processing. IEEE*, 2004, vol. 52, pp. 2788–2797.

[11] C. Shen, W. Plishker, H. Wu, and S. S. Bhattacharyya, "A lightweight dataflow approach for design and implementation of SDR systems," *Proc. Wireless Innovation Conference and Product Exposition*. Wireless Innovation Forum, 2010, pp. 640-645.

[12] E. A. Lee, D. G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, IEEE, 1987, vol. 75, pp. 1235-1245.

[13] "IEEE standard for floating-point arithmetic", *IEEE standard 754-2008*, Aug. 29, 2008.

[14] J. Boutellier, O. Silvén, M. Raulet, "Automatic synthesis of TTA processor networks from RVC-CAL dataflow programs," *Proc. IEEE Workshop on Signal Processing Systems*. IEEE, 2011, pp. 25-30.

[15] Faraday technology, "90nm low-k low leakage regular-Vt standard cells," Product brief v. 1.2.

[16] Faraday technology, "90nm standard performance low-k standard cells," Product brief v. 1.0.

[17] A. Chang, W. J. Dally, "Explaining the gap between ASIC and custom power: a custom perspective," *Proc. 42nd annual Design Automation Conference*. ACM, 2005, pp. 281-284.

[18] D. Baran, M. Aktan, V. G. Oklobdzija, "Energy efficient implementation of parallel CMOS multipliers with improved compressors," *Proc. International Symposium on Low Power Electronics and Design*. ACM, 2010, pp. 147-152.