Hardware Architecture and VLSI Implementation of a Low-Power High-Performance Polyphase Channelizer with Applications to Subband Adaptive Filtering

Yongtao Wang, Hamid Mahmoodi, Lih-Yih Chiou
Hunsoo Choo, Jongsun Park, Woopyo Jeong and Kaushik Roy

School of ECE, Purdue University, West Lafayette, IN 47907, USA

ABSTRACT

Polyphase channelizer is an important component of a subband adaptive filtering system. This paper presents efficient hardware architecture and VLSI implementation of a low-power high-performance polyphase channelizer, integrating optimizations at algorithmic, architectural and circuit level. At the algorithm level, a computationally efficient structure is derived. Tradeoffs between hardware complexity and system performance are explored during the fixed-point modeling of the system. A computational complexity reduction technique is also employed to reduce the complexity of the hardware architecture. Circuit-level optimizations, including an efficient commutator implementation, dual-VDD scheme and novel level-converting flip-flops, are also integrated. Simulation results show that the design consumes 352mW power with system throughput of 480 million samples per second (MSPS). A test chip has been submitted for fabrication to validate the proposed hardware architecture and VLSI design techniques.

1. INTRODUCTION

Subband adaptive filter systems are widely used for adaptive signal processing applications that require filters with very long impulse response and/or suffer from slow convergence [1-6]. In such applications subband adaptive filtering is a viable alternative to conventional least-mean-square (LMS) algorithm because it reduces computational complexity and offers improved convergence rate. In the basic configuration of a subband adaptive filter [2], both input signal x[n] and desired response d[n] are decomposed into M decimated subbands by a polyphase channelizer and all the adaptive filters are operated independently in these subbands. Both update rate and length of the adaptive filters can be greatly reduced resulting in lower computational complexity and power consumption. Further, the processing in separate subbands makes better convergence speed possible in the case of the LMS algorithm, since the adaptation step size in each subband can be matched to the energy of the input signal in that subband [1][2]. Subband signals can be recombined by a polyphase combiner to produce the final output.

The basic structure of a polyphase channelizer is illustrated in Fig. 1, which is essentially a uniform DFT analysis filter bank [7]. The polyphase combiner is the corresponding uniform DFT synthesis filter bank [7].

In the polyphase channelizer, the input signal x[n] is multiplied with the complex exponentials $W_M^{-(k)n} = e^{j2\pi kn/M}, \forall 0 \le k \le M-1$, which is equivalent to a uniform shift in frequency domain. The resulting signals are then passed through a low-pass filter with impulse response h[n], which is usually called prototype filter. The output of the prototype filter is decimated by a factor N≤M to generate each subband signal. The adaptive filtering algorithm and/or other signal processing can then be applied to those subband signals.

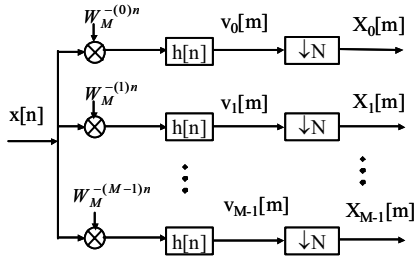


Fig. 1. The basic structure of a polyphase channelizer

When critical sampling is employed (i.e., N=M), the presence of aliasing requires the use of adaptive cross-filters between adjacent subbands [1] or gap filterbanks [4]. However, systems with cross-filters generally converge slower and have higher computation cost, while the distortion produced by gap filterbanks may not be acceptable. Oversampled subband adaptive filtering systems with N<M offer a simplified structure without employing cross-filters or gap filter banks and reduce the alias level in the subbands. In order to reduce the computational cost, the oversampling ratio M/N is usually chosen to be close to one.

To reduce the alias level in the subbands, large prototype filter (e.g., with hundreds of taps) is necessary, which leads to a very high computational throughput requirement and power dissipation, especially when the input data rate is high. Hence, developing efficient VLSI hardware for the polyphase channelizer is very important for the successful deployment of subband adaptive filtering systems. In this work we present hardware architecture and VLSI design techniques for implementing high-performance energy-efficient polyphase channelizer, which are demonstrated through a polyphase channelizer with M=64, N=48. The prototype filter has 768 taps and the input data rate is 480 million samples per second (MSPS).

The rest of the paper is organized as follows. In section 2, a computationally efficient structure is derived for the polyphase channelizer. Section 3 presents the fixed-point modeling and efficient hardware architecture. In section 4, circuit-level techniques are explored to further reduce the power consumption. Final implementation results are presented in section 5 and section 6 concludes the paper.

This research was supported in part by the DARPA MSP program and Semiconductor Research Corporation (1122.001).

2. COMPUTATIONALLY EFFICIENT STRUCTURE OF THE POLYPHASE CHANNELIZER

As shown in Fig. 1, $X_k[m]$ is the output of *k*-th channel, $v_k[n]$ is the output of prototype filter h[n] for *k*-th channel, where $0 \le k \le M-1$. Vector X[m] = $[X_0[m], X_1[m], \ldots, X_{M-1}[m]]^T$ and vector v[n] = $[v_0[n], v_1[n], \ldots, v_{M-1}[n]]^T$.

Since $X_k[m]$ is the decimation of $v_k[m]$ by a factor of N, $X_k[m] = v_k[mN]$, or $X[m] = v[mN]$. And

$$v_k[n] = (x[n]W_M^{-kn}) * h[n] = \sum_{l=-\infty}^{\infty} h[n-l]x[l]W_M^{-kl},$$ where $*$ denotes the convolution operation. It follows that

$$X_k[m] = \sum_{l=-\infty}^{\infty} h[mN-l]x[l]W_M^{-kl} = (\sum_{l=-\infty}^{\infty} h[mN-l]x[l]W_M^{k(mN-l)})W_M^{-kmN}$$
$$= (\sum_{l=-\infty}^{\infty} r_k[mN-l]x[l])W_M^{-kmN} = (r_k[n] * x[n])_{\downarrow N} W_M^{-kmN}$$

where $\downarrow N$ denotes decimation by a factor of N. And $r_k[n] = h[n]W_M^{kn}$, which is, in z-domain, equivalent to $R_k(z) = H(zW_M^{-k})$, where $H(z)$ denotes the z-transform of the prototype filter impulse response h[n], and $W_M^{-k} = e^{j2\pi k/M}$. The filters $R_k(z) = H(zW_M^{-k})$ with $0 \le k \le M-1$, and their following decimation operation form a new filter bank. The input of the filter bank is x[n] and the outputs of this new filter bank are $(r_k[n] * x[n])_{\downarrow N}$, $0 \le k \le M-1$. Let K be the least common multiple of M and N, and let J and L be the two integers satisfying K = JM = LN. Through polyphase decomposition and factorization, this new filter bank can be transformed into three serially connected processing blocks, i.e., a commutator with decimation factor of N, an M×N polyphase matrix and an M×M DFT matrix [8]. Consequently a computationally efficient structure is derived as shown in Fig. 2. The commutator is composed of a delay chain followed by decimators. The elements of B(z) are given by

$$[B(z)]_{ij} = z^{-l}Q_{(j+lN)}(z^L) \qquad (i-j) \bmod g = 0$$
$$= 0 \qquad (i-j) \bmod g \ne 0$$

$$(j+lN) \bmod M = i, g = \gcd(M,N), 0 \le i \le M-1, 0 \le j \le N-1$$

where gcd(M,N) is the greatest common divisor of M and N, and $Q_l(z)$ is the *l*-th K-th order polyphase component of the prototype filter h[n].

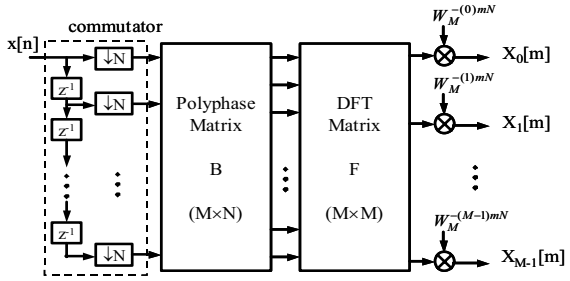


Fig. 2. Computationally efficient polyphase channelizer

In this structure, excluding the commutator, the whole system is operating at an N times lower rate than the input data rate, resulting in considerably lower computational complexity. Instead of requiring one large prototype filter h[n] for each channel, for all M channels only one polyphase matrix B is needed. DFT matrix can be efficiently implemented using the FFT structure. For M=64, N=48, multiplication with the complex exponential W_M^{-kmN} is trivial since W_M^{-kmN} reduces to $(-j)^{km}$.

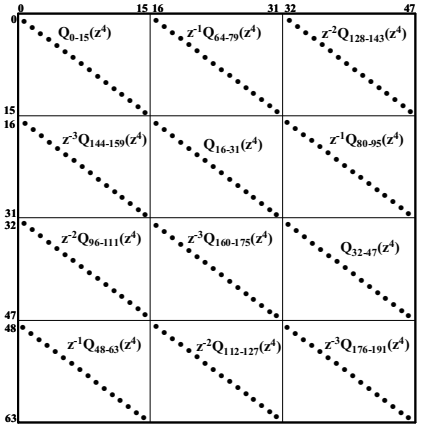


Fig. 3. Structure of the polyphase matrix B with M=64, N=48

Structure of the polyphase matrix B with M=64 and N=48 is shown in Fig. 3, in which each non-zero element is represented by a dot. There are totally K=192 nonzero elements in B. The whole matrix B can be divided into 12 16×16 sub-matrices. The nonzero elements of B are located on the main diagonal of these sub-matrices. For example, in the second sub-matrix on the first row, the non-zero elements on its main diagonal are shown in Fig. 3 as $z^{-1}Q_{64-79}(z^4)$, i.e. the first non-zero element along the diagonal is $z^{-1}Q_{64}(z^4)$, the second non-zero element along the diagonal is $z^{-1}Q_{65}(z^4)$ and so on. All other non-zero elements are similarly illustrated in Fig. 3.

3. FIXED-POINT MODELING AND EFFICIENT HARDWARE ARCHITECTURE

In the fixed-point modeling of the system, there exists a fundamental trade-off between hardware complexity and system performance. In terms of hardware complexity our major consideration is to determine the bit-lengths of several system parameters and important signal nodes. Judicious choice of these bit-lengths has a significant impact on hardware complexity and system performance.

Integrated side-lobe ratio (ISLR) is chosen as the metric for system performance. ISLR is defined as the ratio of the energy in the stopband to the energy in the passband of the prototype filter. The *signal-to-alias* ratio (SAR) defined in [6] can be approximately expressed as $1 + \frac{1}{ISLR}$. Hence a small ISLR is desirable for subband adaptive filtering systems to achieve a small minimum mean-square error (MMSE) [6].

We start by exploring the effect of the bit-length of the filter coefficients on ISLR. Relationship between bit length of prototype filter coefficients and ISLR is shown in Fig. 4. When the bit length is greater than 13, ISLR levels off and using more bits has little or no impact on ISLR. There exists a theoretical limit on achievable ISLR value for a given prototype filter. In the region where bit length is smaller than 13, ISLR takes off and small reduction on bit length will lead to a large increase in ISLR. Hence the optimal point is the *knee* of the curve, which corresponds to a bit length of 13 and an ISLR of –73.4dB.

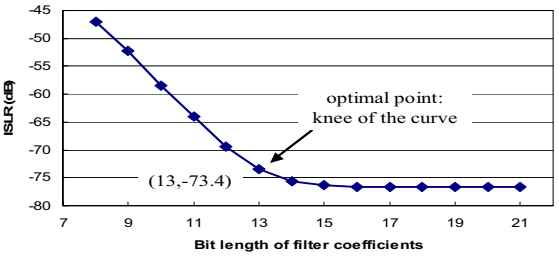


Fig. 4. ISLR vs. Bit-length of filter coefficients

The polyphase matrix B(z) represents an MIMO (Multi-Input Multi-Output) system. The inputs to B(z) are denoted as vector $A(z) = [A_0(z), A_1(z),..., A_{47}(z)]$ and outputs of B(z) are denoted as vector $U(z) = [U_0(z), U_1(z),..., U_{63}(z)]$. It follows that $U(z) = B(z)A(z)$. Since there are four nonzero elements in each column of polyphase matrix B, every input of B(z), namely $A_k(z)$, is multiplied by 4 polyphase filters: $Q_k, Q_{k+48}, Q_{k+96}, Q_{k+144}$. Since the prototype filter has 768 taps, each polyphase filter $Q_l(z)$ has 768/172=4 taps. If these polyphase filters are implemented in the transposed form, then every input of B(z) is multiplied by 16 filter coefficients. Computational complexity reduction on these multiplications has a large impact on the hardware implementation of the polyphase matrix B. We developed an efficient computational complexity reduction technique, called Computation Sharing Differential Coefficient (CSDC) method [10], which can be used to obtain low-complexity parallel multiplierless implementation of FIR filters and DSP tasks involving multiplications with a set of constants. Compared with the implementation in which all the coefficients are encoded in the canonical signed digit (CSD) format, 57% complexity reduction in terms of the number of additions has been achieved by using the CSDC technique, resulting in smaller area, lower power consumption and higher speed.

Since M=64, for implementing the DFT matrix, we use the well-known radix-4 FFT structure, which has three stages. The important nodes under consideration include the output nodes of the polyphase matrix, output nodes of the first and second stages of the FFT and the final outputs. Outputs of the final stage of the FFT have the same bit-length as the final outputs since magnitude of the complex exponential, W_M^{-kmN}, is 1. Fixed-point modeling and extensive simulations in Matlab and Simulink have been performed to determine the bit-length of DFT twiddle factors and the important nodes mentioned above. Appropriate scaling is also employed to avoid overflow. Within the polyphase matrix, for the given prototype filter, the maximum possible gain is about 1.85, which is less than 2. Hence scaling by 0.5 is enough to avoid overflow. Scaling by 0.25 is applied to output of each stage of the FFT. Table 1 summarizes the bit-lengths used for final hardware implementation. The resulting ISLR is –66dB.

Table 1. Summary of the bit-lengths used for hardware implementation

Filter Coefficients	Outputs Of B	FFT Twiddle Factors	Outputs of 1st and 2nd FFT stage	Final outputs
13	16	16	16	16

4. CIRCUIT LEVEL TECHNIQUES

4.1 An efficient commutator circuit implementation with Double Data Rate (DDR) data input

The input data (DATA) of polyphase channelizer are generated by an ADC (analog-to-Digital Converter) and are fed into the polyphase channelizer using a Double Data Rate (DDR) format with a DATA_VALID signal. The assertion of DATA_VALID signal indicates that the input data is valid. The timing diagram is illustrated in Fig. 5.

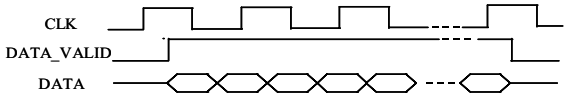
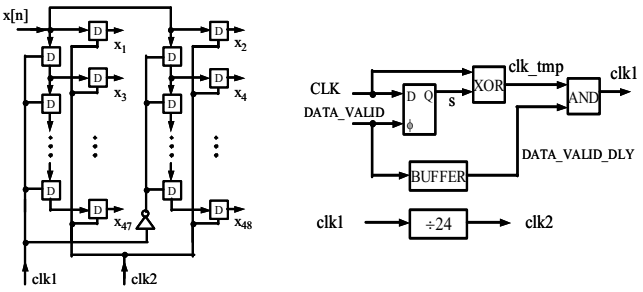


Fig. 5. Timing diagram of DDR data input

As shown in Fig. 2, the commutator is composed of a delay chain, which consists of 47 serially connected delay elements, and 48 decimators. The outputs of the commutator are denoted as $x_1[m]$, $x_2[m]$, …, $x_{47}[m]$, $x_{48}[m]$, from top to bottom. One straightforward implementation is to use dual-edge triggered flip-flops. However, using dual-edge triggered flip-flops would incur larger area and power consumption than single-edge triggered flip-flops.

We developed an efficient commutator implementation, which uses only positive-edge triggered flip-flops, and a clock generation circuit as shown in Fig. 6 (a) and (b), respectively. As shown in Fig. 6 (a), the input data sequence is first broken into two sequences: an odd data sequence and an even data sequence. Then these two data sequences are sampled by the flip-flops driven by clock signal clk2. However, in order to make the proposed circuit work, generating appropriate clock signals clk1 and clk2 is critical.



(a) Commutator circuit (b) Clock generation circuit

Fig. 6 Efficient commutator implementation

When DATA_VALID goes from low to high, first sampling edge of clk1 shall be a rising edge. Otherwise input data will not be correctly sampled. Clock signal CLK cannot meet this requirement since the edge of CLK right after the rising edge of DATA_VALID can be either a rising edge or a falling edge. The clock generation circuitry is shown in Fig. 6(b). In the generation of clk1, first we use the rising edge of signal DATA_VALID to sample signal CLK and generate signal **s** (Note that initially **s** will be reset to low). If **s** is high, the first sampling edge of CLK must be a falling edge. Otherwise the first sampling edge of CLK must be a rising edge. Propagating signal s and CLK through an XOR gate generates the signal clk_tmp. As a result, the first sampling edge of CLK will generate a rising edge on clk_tmp, which can be used to sample the first input data sample. To remove possible glitches on signal clk_tmp, signal DATA_VALID is delayed by a buffer (BUFFER), generating signal DATA_VALID_DLY.

Finally DATA_VALID_DLY signal and clk_tmp go through an AND gate such that the glitches in signal clk_tmp will not propagate through to clk1, thus producing a glitch-free clock signal clk1. It is necessary to carefully adjust the delay of the buffer (BUFFER) to make sure that clock signal clk1 is generated as desired. Dividing clk1 by 24 produces the clock signal clk2. Consequently, in conjunction with the clock generation circuit, the commutator is efficiently implemented without using dual-edge triggered flip-flops. Given the fact that the commutator is operating at a data rate 48 times as high as the rest of the polyphase channelizer, this efficient implementation of the commutator leads to a considerable amount of power saving.

4.2 Dual-VDD scheme and level-converting flip-flops

As pointed out in section 2, the whole system except the commutator operates at a rate 1/48 of the input data rate. Therefore, we can apply the nominal supply voltage to the commutator while the supply voltage of the rest of the system (i.e. polyphase matrix, FFT and multiplications with the complex exponentials W_M^{-kmN}) can be scaled down. Due to the quadratic dependence between the switching power and supply voltage, such a dual-VDD scheme can lead to significant power savings. The associated overhead is the level conversion that is required to raise the output signal level to the high supply voltage at the interface from the low-VDD block to the high-VDD block.

To export the computation results of the subbands, multiplexing is usually employed. The multiplexer usually consists of two stages. At the first stage all the channel outputs are sampled and latched into flip-flops. At the second stage the latched data are serially sent off the chip. Conventionally the level converters and the flip-flips in the first stage of the multiplexer are designed and optimized separately. In this work we develop a new circuit, Level-Converting Flip-Flop (LCFF) [9], which merges a level converter and a flip-flip, leading to reduced area and power consumption and higher performance. This flip-flop is a pulsed flip-flop that employs conditional capturing and self-precharging techniques to efficiently perform latching and level converting functions [9].

5. CHIP IMPLEMENTATION AND RESULTS

We developed an integrated and well-automated design flow from algorithmic optimization and fixed-point modeling in Maltab/Simulink, VHDL coding and logic synthesis to physical design. Layout of the LCFF is custom designed and optimized. The rest of the design, including the commutator, the clock generation circuit, polyphase matrix, FFT and multiplexer are coded in VHDL, synthesized using Synopsys tools and their layouts are separately generated in Silicon Ensemble. Since our design incorporates different VDD and combines custom and semi-custom blocks, the final layout is generated by assembling the layouts of all the constituent blocks in IC Craftsman. We used 0.18μm CMOS technology. When the whole system is operating at the nominal supply voltage of 1.8V (i.e. without employing dual-VDD scheme), the total power consumption is 844mW with a throughput of 480MSPS. However, using the proposed dual-VDD scheme, the lower VDD can be 0.9V and this leads to a power consumption of 352mW. The total area is of the chip is about 64mm^2.

A test chip has been implemented and submitted for fabrication to validate the proposed hardware architecture and VLSI design techniques. Its layout is shown in Fig. 7.

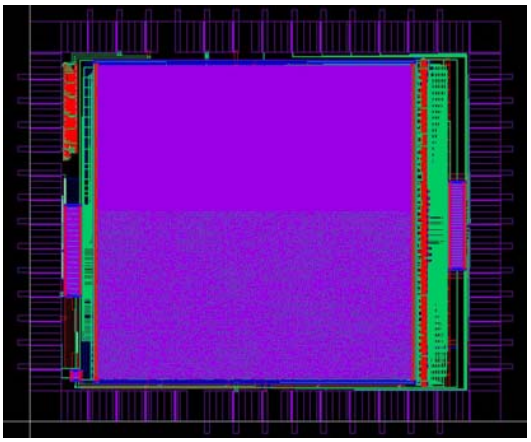


Fig. 7. Layout of the test chip

6. CONCLUSIONS

We presented efficient hardware architecture and VLSI implementation of a low-power high-performance polyphase channelizer. Optimizations at the algorithmic, architectural and circuit level are integrated to achieve high system throughput with low power consumption. A total power consumption of 352mW is achieved with a throughput of 480MSPS. A test chip has been submitted for fabrication to validate the proposed hardware architecture and VLSI design techniques.

7. REFERENCES

[1] A. Gilloire and M. Vetterli, " Adaptive filtering in subbands with critical sampling: Analysis, experiments and applications to acoustic echo cancellation," *IEEE Trans. On Signal Processing,* vol. 40, pp. 1862-1875, Aug. 1992.

[2] J. J. Shynk, "Frequency-domain and multirate adaptive filtering", *IEEE Signal Processing Magazine*, vol. 9, pp. 14-37, Jan. 1992.

[3] S. Weiss, et al, "Adaptive equalization in oversampled subbands," *Electronics Letters,* vol. 34, no. 15, pp. 1452-1453, July 1998.

[4] O. Tanrikulu, et al, "Residual echo signal in critically sampled subband acoustic echo cancellers based on IIR and FIR filter banks." *IEEE Trans. On Signal Processing*, vol. 45, no. 4, pp. 901-912, 1997.

[5] W. S. Song, et al, "High-performance low-power polyphase channelizer chip-set", *Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1691-1694, 2000.

[6] S. Weiss, et al, "Steady-state performance limitations of subband adaptive filters*," IEEE Trans. On Signal Processing*, vol. 49, no. 9, pp. 1982-1991, September, 2001.

[7] J. G. Proakis and D. G. Manolakis, *Digital signal processing: principles, algorithms and applications*, Third edition, Prentice Hall Inc., 1996.

[8] Z. Cvetkovic and M. Vetterli, "Tight Weyle-Heisenberg frames in $\ell^2(Z)$ ", *IEEE Trans. Signal Processing*, vol. 46, no. 5, May 1998.

[9] H. Mahmoodi-Meimand and K. Roy, "Self precharging flip-flop (SPFF): a new level-converting flip-flop," *European Solid-State Circuits Conference*, pp. 407-410, Sep. 2002.

[10] Y. Wang and K. Roy, "CSDC: a new complexity reduction technique for parallel multiplierless implementation of digital FIR filters", submitted to *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing.*