Low Power Techniques and Design Tradeoffs in Adaptive FIR Filtering for PRML Read Channels

Khurram Muhammad[1], Robert B. Staszewski[1] and Poras T. Balsara[2]
(k-muhammad1@ti.com, b-staszewski@ti.com, poras@utdallas.edu)
[1]Texas Instruments Inc, Dallas, TX 75243, USA
[2]Department of Electrical Engineering, Univ. of Texas at Dallas, Richardson, TX 75083, USA

ABSTRACT
In this paper, we describe area and power reduction techniques for a low-latency adaptive *finite-impulse response* filter for magnetic recording read channel applications. Various techniques are used to reduce area and power dissipation while speed remains as the main performance criterion for the target application. A parallel *transposed direct form* architecture operates on real-time input data samples and employs a fast, low-area multiplier based on selection of radix-8 pre-multiplied coefficients in conjunction with one-hot encoded bus leading to a very compact layout and reduced power dissipation. Area, speed and power comparisons with other low-power implementation options are also shown. The proposed filter has been fabricated using a 0.18 μm L-effective CMOS technology and operates at 550 MSamples/s.

1. INTRODUCTION
Partial response maximum likelihood (PRML) equalization of magnetic recording read channels [1] is the recent breakthrough in magnetic storage technology and is widely used in commercial hard-disk drives. In this technique, spectral shaping of read back signal is performed using a combination of a continuous time and a digital *finite-impulse response* (FIR) filter [2]. Using the Viterbi algorithm, the most likely symbol sequence is detected on a trellis which results due to the spectral shaping operation. The coefficients of the FIR filter are adapted to provide a desired channel response typically using the *least mean square* (LMS) algorithm. Efficient timing recovery in a read channel is critical for fast phase and frequency acquisition in addition to acceptable *bit error rate* performance. Therefore, it is also critical that the discrete-time spectral shaping filters are implemented with as little latency as possible since the output of these filters is used to extract timing information.

In a typical PRML read channel, the FIR filter may take up to 15% of the total chip area. Storage capacity of media typically doubles every eighteen months, and therefore, faster data retrieval rates are consistently needed. This requires more aggressive techniques for every new design. At the same time, new features are required which inevitably increase the total area of the read channel. Consequently, power dissipation is becoming one of the major design concerns in modern read channels. For a chip with extremely high volume, cost is another major concern and lower area designs are very important for both power and cost. Power dissipation also compounds the cost problem if more expensive packaging is required. Hence, in order to meet the challenge of fast data transfer rate, new architectural and circuit design approaches are required which provide high-speed operation with low area and low power dissipation.

In this paper, we combine many architectural and circuit design techniques to obtain a fast, low-area and low-power adaptive FIR filter. The proposed architecture uses a novel parallel structure which increases the operational speed by a factor of two while keeping the overall increase in area to less than this factor. Fast radix-8 multiplication is accomplished using a select and add of pre-multiplied coefficients. This scheme is considerably simpler and lower area than using a conventional multiplier and more effective than pipelined *direct form* (DF) implementations. The proposed filter is compared with other implementation options to demonstrate that the proposed scheme achieves the best speed-area-power tradeoff for the application.

ISLPED '00, Rapallo, Italy.

2. GENERAL IMPLEMENTATION TECHNIQUES
In this section, we will consider various adaptive FIR filter implementation approaches for the read channel. The main design goals in this application are high speed, low latency, low area and low power — all conflicting requirements. Therefore, the design requires intelligent choices which result in best implementation for the application. It is generally believed that DF implementation results in lower area and lower power dissipation while *transposed direct form* (TDF) offers higher speed but requires larger area. In this paper, we will show that when speed and latency targets cannot be compromised, DF implementations are not viable alternatives — even when pipelining and parallel processing is used to improve the speed of operation. We also show that in applications where speed and latency are not critical, TDF offers better speed-area-power performance than the DF implementations.

2.1 TDF Implementation with Conventional Multiplication
This scheme uses a radix-4 Booth-encoded Wallace tree multiplier in the filter implemented in TDF and will be referred to as *TDF-BWT* implementation. As shown in Fig. 1, the critical path consists of a multiply-and-add operation in each stage. The carry-save format of the multiply-and-add output at each stage is converted to regular binary format. This nearly halves the number of registers

required for storing intermediate results in contrast to storing carry-save outputs. Further, it reduces the latency of the filter since no final carry propagation is needed beyond the last stage. For high-speed designs which push the technology to its limits, registers with very small *CLK-to-Q* delay, setup and hold times are required. This increases the area and power dissipation in registers and, therefore, reduction of registers is highly desirable.

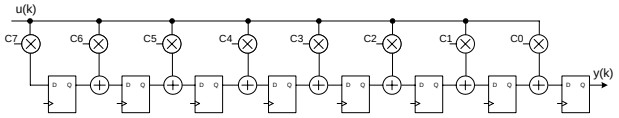

Figure 1: TDF FIR Implementation.

2.2 DF Implementation without Pipelining

Fig. 2 shows the block diagram of a DF FIR filter without pipelining. We will refer to this implementation as *DF-Pipe-0* implementation. This implementation is generally considered to be the lowest area implementation, however, it suffers from speed disadvantage. The critical path consists of the *CLK-to-Q* delay and set-up time of the storage register, one multiplication and $O(logN)$ addition stages, where N is the number of filter taps. The multipliers used in this implementation are radix-4 Booth-encoded Wallace tree multipliers. The output of the multipliers are kept in carry-save format and added using an adder tree comprising full adders as shown in figure. The final carry-save output is converted to regular binary-format using a vector merge stage.

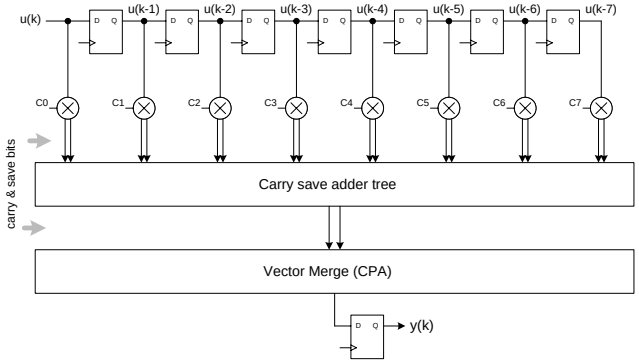

Figure 2: DF FIR Implementation — no pipelining.

2.3 DF Implementation with 1 Pipeline Stage

This implementation will be referred to as *DF-Pipe-1* implementation. The block diagram of this implementation is shown in Fig. 3 where one pipelining stage is inserted between the multipliers and the adder tree. Again, Booth-encoded Wallace tree multipliers are used and their output is converted to regular binary format. We noted that registering multiplier outputs in carry-save format not only doubles the area and power-intensive pipelining registers, but it also increases the number of operands in the adder tree thereby increasing the delay in the next stage. The carry-save adder tree output is converted to regular binary format using a vector merge stage. The delay due to vector merge prior to pipelining registers did not increase the length of the critical path, as it is dominated by the adder tree. In this case, the critical path consists of the *CLK-to-Q* delay, set up and hold time of the storage registers plus the maximum of the multiplier and the adder-tree delays. The latency of this filter implementation is equal to the latency of the proposed architecture.

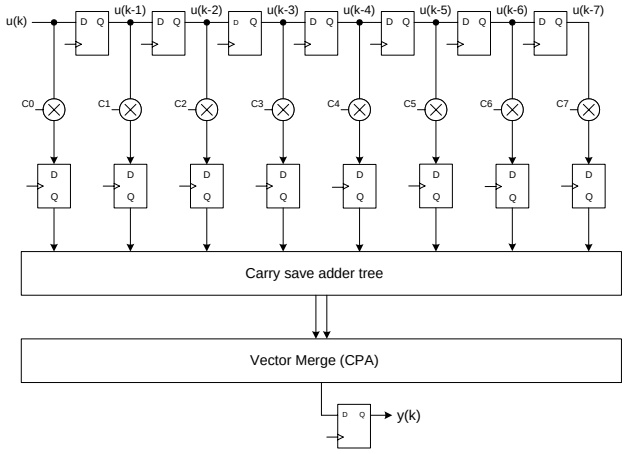

Figure 3: DF FIR Implementation with 1 pipeline stage.

2.4 DF Implementation with 2 Pipeline Stages

Fig. 4 shows the block diagram of the direct form filter implementation with two pipelining stages and will be referred to as *DF-Pipe-2* implementation. This filter implementation has one higher latency than the proposed architecture which is traded-off for an increase in the operating speed. As shown in figure, the first pipelining stage is inserted at the output of the multipliers which are kept in carry-save format for reducing the worst case delay. The carry-save output of the adder tree is registered in the second stage of pipelining registers. The final output stage converts the carry save output to 14-bit regular binary format using a vector merge stage. In this architecture, the critical path consists of the *CLK-to-Q* and set-up times of the storage registers plus the maximum of the delays through any of the pipeline stage. The largest delay was observed in the adder tree which was required to add 16 binary numbers (i.e., eight carry-save outputs).

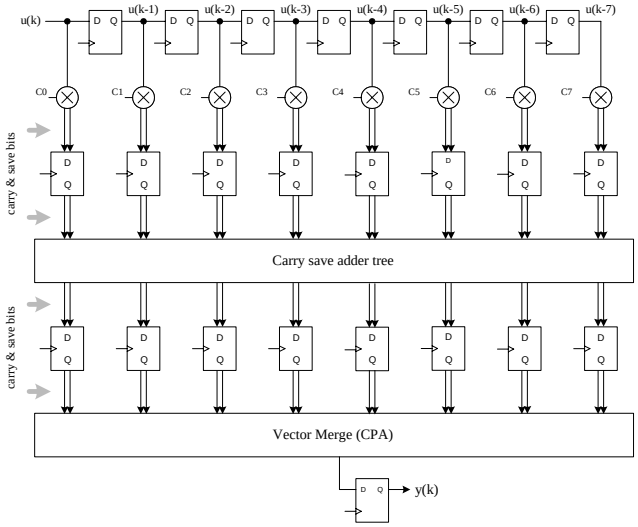

Figure 4: DF FIR Implementation with 2 pipeline stages.

2.5 Interleaved DF implementation

Any DF implementation can be interleaved [3] to provide faster speed of operation using the principle of parallel processing. In this scheme, the data stream can be replicated such that in one stream

the even data leads the odd data while in the second stream, the odd data leads the even data. Both streams are staggered in time by one clock cycle with respect to each other and each stream is operated upon by an independent filter. Two filters are required for an interleaved design with two data streams and output two samples every clock cycle. We will consider interleaved DF filter implementations for each of the pipelined DF filters described earlier.

3. PROPOSED TRANSPOSE-TYPE ARCHITECTURE

In this section, we will describe the proposed architecture. The design choices made in the proposed architecture attempt to provide the best operating point in the speed-area-power-latency space. The impact of choices made in the presented implementation will become apparent in section 4 where this architecture is compared with other implementations presented in section 2.

3.1 Parallel Architecture

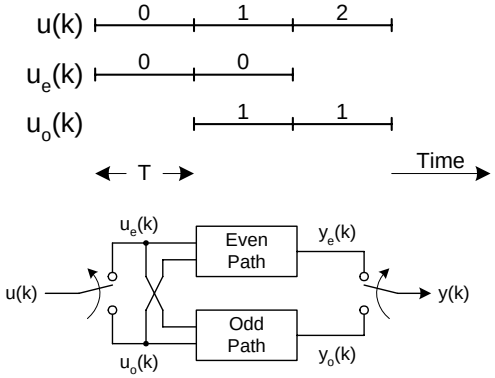

Figure 5: Parallel FIR filter operation.

Let $y(k)$ represent the output of an N-tap FIR filter at time instant k. Then $y(k) = c_0 u(k) + c_1 u(k-1) + ... + c_{N-1} u(k-N+1)$, where c_n and $u(k)$ represent the nth coefficient and the input data sample at time instant k, respectively. This operation could be performed in parallel as shown in Fig. 5. Fig. 6 shows the basic idea of parallel TDF architecture. This structure is derived for real-time data where $u_e(k)$ is the even interleave and appears at the upper input at time slot k but is stable during the following odd time slot $k+1$. The data $u_o(k)$ is an odd interleave and arrives at the lower input at odd time slot k but is stable during the following even time slot $k+1$. The FIR operational speed is doubled since *multiply-and-add* operation is now performed at half the data rate. An important advantage of this structure is that it allows computation sharing amongst the respective even and odd multiply operations in the two parallel paths. By using a numbering system that is higher than radix-2, some operations can be made common to both parallel paths.

This new architecture naturally allows the application of latches in the internal clocking stages, as a faster, smaller and lower-power alternative to using flip-flops. The proposed architecture takes advantage of the normal irregularity of critical path delays between neighboring stages by borrowing timing slacks from the less time-critical taps. As a result, the operational throughput could be more than doubled with less than twice hardware cost and area. This is because the register overhead for the desired application was found to be 33% of the clock period. However, by using the proposed

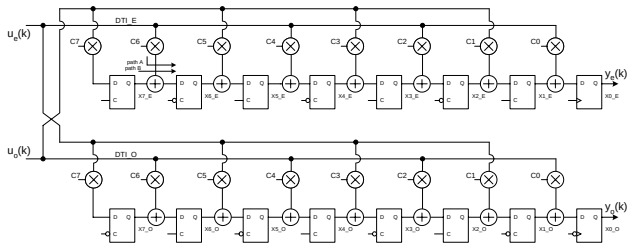

Figure 6: Parallel TDF type 8-tap FIR filter structure.

structure, this overhead reduces to 16.5% of the half-rate clock and allows speed up by a factor slightly greater than two.

This is in sharp contrast to pipelining which is normally used to achieve higher speed. Pipelining also adds latency, area and power overhead since extra latching or re-clocking stages for every pipelining order are required. These stages also add complexity to the clock tree. In addition, the circuit is still clocked at the same high frequency as the data rate which increases the dynamic power dissipation. Our scheme alleviates these problems without the need of input buffering as the even and odd data samples are applied at the respective inputs exactly at the time when they are available.

3.2 Low Area Radix-8 Booth Architecture

In the proposed architecture we encode the incoming high-speed 6-bit data into radix-8 numbering system. The main advantage of radix-8 encoding of data is that the 3x coefficient pre-multiplication is performed off the critical path.

3.2.1 Data Encoding versus Coefficient Encoding

Encoding data allows reduction of area by sharing of resources. Fig. 7 shows the basic concept. The physical format of the encoded data for each of the parallel paths consists of two buses: The first bus is a collection of 9 wires and is a function of the higher-order 4 bits of the original input data. The second bus is one-wire smaller, 8-bit wide, and is a function of the lower-order 3 bits of the original input data. One bit is shared between the two encoded numbers and leads to a redundant arithmetic system. The bits within each bus are encoded in one-hot manner, meaning that at all times an exactly one bit is asserted. This reduces the power dissipation as well as area since both buses run straight to all taps of the FIR filter resulting in a regular and compact layout (see Fig. 8).

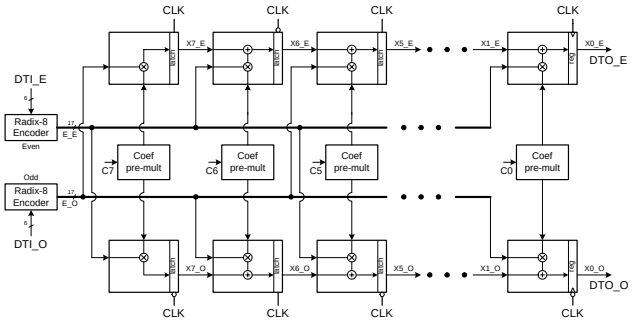

Figure 7: Parallel 8-tap FIR filter with radix-8 encoding of input data.

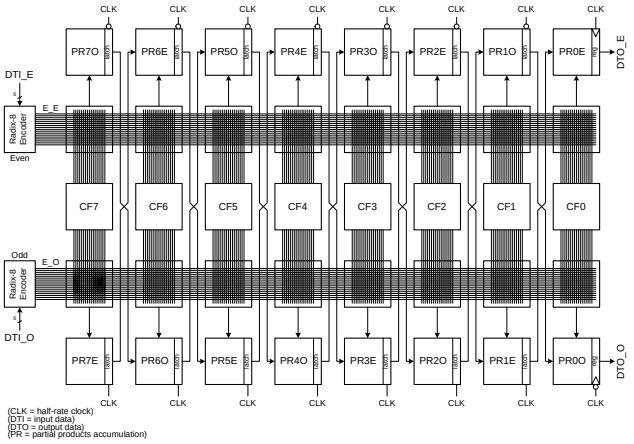

Figure 8: Floorplan of the 8-tap FIR filter.

3.2.2 Low-power Pre-multiplication

Each FIR coefficient is pre-multiplied for the following cases: -4C, -3C, -2C, -C, 0C, C, 2C, 3C, 4C, where C is the coefficient value. The 0C (zero) and power-of-two pre-multiplications are trivial. Similarly -2C and -4C cases are a simple left shift operation of the pre-negated -C coefficient. As a result, only the negation (-C) and multiplication-by-three (3C) non-trivial operations are required. The multiplier structure is shown in Fig. 9 where a multiplexer shown in Fig. 10 selects the appropriate pre-multiplied coefficient. Since the FIR coefficients in read-channel equalization do not usually change at high rate, the precomputation does not require the high-speed operation of coefficients pre-multiplication. Hence, the critical path of the multiplier does not include the delay in pre-multiplication. The minimum size NMOS–based multiplexer cell in Fig. 10 has a compact layout and features a low average switched capacitance. This significantly reduces power while allowing the cell to operate at high speed. The combination of one-hot operation of each bus with the above pass-gate based multiplexer significantly reduces the average switched capacitance and results in a fast and low-power multiplier.

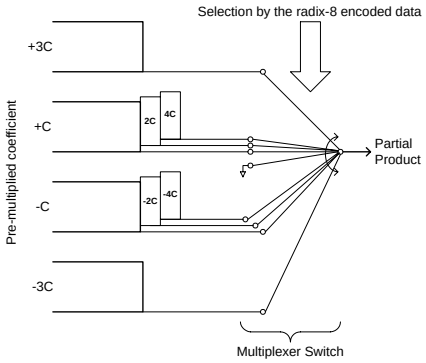

Figure 9: Radix-8 multiplier based on selection of pre-multiplied coefficients.

In case faster pre-computation of FIR coefficient is desired, the coefficients pre-multiplication operation could be easily retimed through pipelining. The resulting coefficient update latency of one or two clock cycles is negligible as compared with the slow rate of the LMS adaptation itself.

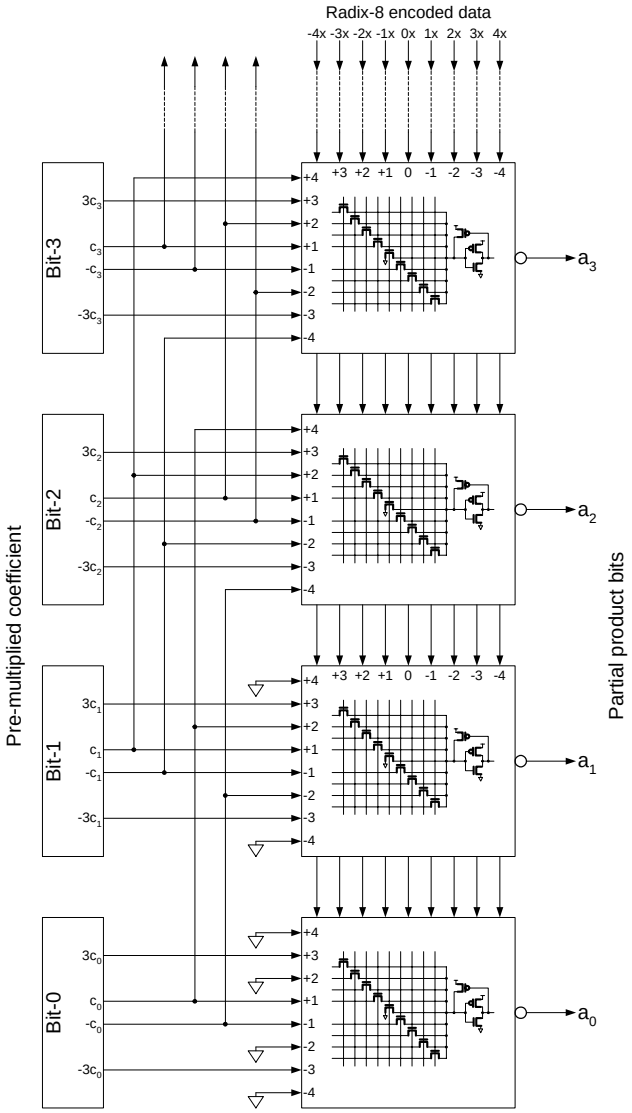

Figure 10: Radix-8 multiplier built with low-area multiplexer cell array.

3.3 Efficient Quantization

A rounding scheme has been implemented that reduces hardware complexity without significantly affecting the system performance. It uses the idea of playing off a single large negative average error due to truncation versus a smaller positive average error due to rounding contributed over multiple taps — such that the resulting average DC offset at the output is very close to zero. It has been verified through system simulation (see [8]) that the RMS error attributed to this rounding scheme (with the bias component removed) was below $1/2$ of the output LSB. The basic idea is shown in Fig. 11. All the coefficients have the same resolution of $1/32$. Consequently, the weight of an internal LSB bit corresponds to the 2^{-5} weight of the external LSB. The three-bit rounding is performed identically at each of the eight stages and is realized as a truncation of the partial product 2^0 and 2^1 bits followed by rounding off of the 2^2 LSB bit of the accumulated sum. The final truncation of two bits (2^3 and 2^4 internal LSB weight) is performed just before the filter's output Y at the zero*th* tap (i.e. $X0$). This

Table 1: Area, speed and power dissipation comparison of the proposed filter with other alternatives.

Filter Type	Speed (Msps)	Area/ Tap	P_{DISS} (mW)	L at	Msps/ Area/W	Msps/ mW
Proposed	550	1125	36	2	13.6	15.3
TDF-BWT	350	1009	84	1	4.1	4.2
DF-Pipe-0	220	769	42	1	6.8	5.2
DF-Pipe-1	337	1027	80	2	4.1	4.2
DF-Pipe-2	423	973	108	3	4.0	3.9

contributes to the compensating negative bias.

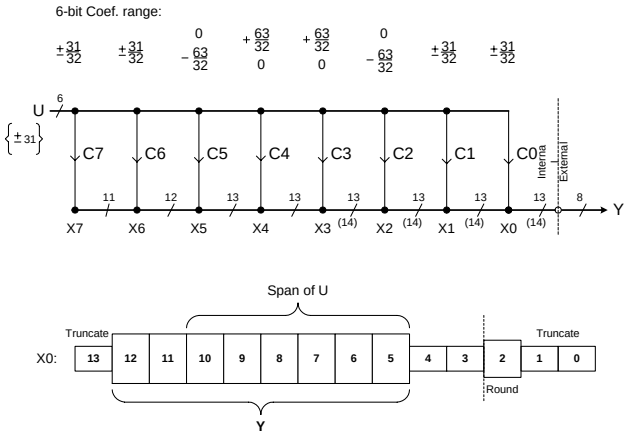

Figure 11: Bit resolution at various tap positions and rounding scheme.

4. COMPARISON OF VARIOUS APPROACHES

This section highlights some salient advantages of the proposed architecture by comparing it with the alternative implementations. Table 1 compares the area (in equivalent gates), speed and power of the proposed filter with the other candidates. The power number of the proposed design were extracted using Powermill. The power dissipation results for all the contending designs were extracted using Synopsys Design Compiler by interpolating from the dynamic power dissipation lookup tables of the cell library. Entries into the two-dimensional lookup table are selected based on the gate input transition time and the gate output load. Thus obtained transition energy of each gate is multiplied by its switching activity.

When compared to TDF-BWT, the proposed architecture slightly increases the area while increasing the speed by 60%. This shows the effectiveness of the proposed multiplication scheme which results in a faster, low-area and low-power alternative to the traditional Booth-encoded Wallace tree multiplier. The relative speed improvement of the precalculation-based multiplier is significantly higher as the 60% improvement is achieved despite the *CLK-to-Q* delay and the set-up time of the registers as well as the delay of the add operation. This is due to reduced area and faster critical path in the proposed scheme.

In Table 1, the entries for the direct form implementations are obtained for filters that do not use any interleaving. Hence, the maximum operating speed of these can only be increased using pipelining. Pipelining overhead becomes apparent when we consider the figure of merit Msps/Area/W, or a better-known inverse of the power-delay product Msps/mW. Observe that putting one pipelining stage only improves the throughput by a factor of 1.5. Using two pipelining stages improves the throughput by another factor of 1.25. This is because it is harder to "equalize" the delays through different pipelining stages as the number of pipelining stages increase. Further, in a design such as an FIR filter, one must remember that moving the pipelining stage in an adder tree can result in a significant increase in pipelining registers, thereby exploding the area and power dissipation. The insertion of pipelining registers in the direct form filters was based on most effective increase in speed without an explosive area growth. Although the critical path in DF-Pipe-2 resided in the adder tree, any attempt to move the second level of pipelining registers to decrease this delay resulted in a massive increase in registers. The clock cycle latency of each filter solution is also shown in Table 1 and abbreviated by *Lat*. As explained earlier, smaller latency is highly desirable for more agile timing recovery loop as the filter output is used to extract timing information. Figure 12 shows the typical nesting of timing recovery and filter adaptation loops in the read channel.

The results shown in Table 1 deserve further elaboration. For a filter implemented for highest possible performance using the given technology, the area and power consumption in registers and other cells increases tremendously as compared to a design operating at a much lower speed. This is because every technology has a "sweet-spot" with optimum area-speed-power operating point for each cell in the library. When the design constraints push for highest speeds possible using the given technology, the area of cells increase rapidly while providing only marginal speed advantage. In general, after selecting a technology, a library of cells is constructed which offer multiple alternatives to a function providing various area, speed and power dissipation operating points. However, in an application operating at highest possible speed, the cells with highest speed are inevitably selected. The area of the fastest register, for example, may be more than twice the area of a reasonably fast alternative. Hence, higher speed design using architectural changes can be a effective approach since it may allow selection of slower, but much smaller cells, thereby tremendously impacting the area and power dissipation. Hence a fair comparison of different architectural techniques requires them to operate at a common speed.

To appreciate this point further, consider the direct form implementations which are designed to operate at maximum possible data rate which is a constraint in speed critical application such as the read channel. The critical path in each of the direct form implementations require use of the fastest possible storage elements and other cells. This has a tremendous impact on the overall area of the implementation, while each still falls short of the speed achievable using the TDF. One could use an interleaved design with the direct-form implementations which provides twice as high frequency of operation in each case while doubling the area and the power dissipation.

Table 2 demonstrates this point by providing a comparison of the proposed filter with interleaved direct form implementations designed to operate at 550 Msps. In this scheme, both pipelining and parallel processing are employed to obtain fast-enough direct form implementation with pre-determined number of pipelining stages. The number of interleaves are shown in the second column. Again, looking at the Speed/Area/P_{DISS} figure of merit, we recognize that the proposed architecture offers the best operating point for speed, area and power dissipation. In an application with very high volume of number of devices, it is imperative that good compromise is

Table 2: Comparison of the proposed filter with interleaved DF implementations for the target speed of 550 Msps.

Filter Type	#Int	Speed (Msps)	Area/ Tap	P_{DISS} (mW)	Msps/ Area/W	Msps/ mW
Proposed	2	550	1125	36	13.6	15.3
DF-Pipe-0	3	550	2006	84	3.3	6.5
DF-Pipe-1	2	550	1575	108	3.2	5.1
DF-Pipe-2	2	550	1790	126	2.4	4.4

Table 3: μW/Msps/Tap/Inbits/Coeff-Bits figure of merit [Thon].

Paper, Implem.	Gate (μm)	η	P_{DISS} (mW)	Speed (Msps)	Area (mm^2)	#Int
This, TDF	0.18	0.23	36	550	0.3	2
[4], TDF	0.8	6.16	426	240	2.9	1
[5], DF	1.2	4.86	50	50	36.4	4
[6], DF	0.8	4.86	140	100	5.85	1
[7], DF	0.7	6.25	165	200	1.1	2

obtained for speed, area and power dissipation and the best choice is very application specific.

The proposed architecture has been implemented in a 0.18 μm L_{eff} technology using a commercial CMOS standard cell [9] digital flow methodology. It is part of a commercial read channel and the die-micrograph of the filter area is shown in Fig. 13. Table 3 compares the proposed filter by earlier reported work. Except for [7], each work implemented an 8-tap FIR filter with 6-bit coefficient and data. [7] reported using an average of 4.4 bits per coefficient. In this table η represents μW/Msps/Tap/InBits/Coeff-Bits.

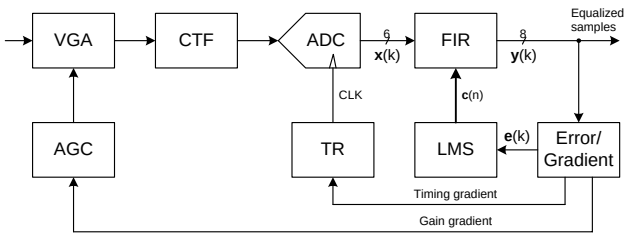

Figure 12: Timing recovery, AGC and FIR filter adaptation loops in a read-channel. Smaller FIR latency improves the

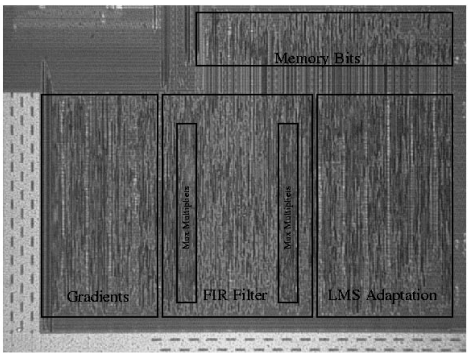

Figure 13: Chip micrograph of the FIR filter area.

5. CONCLUSION

We presented a high-speed, low-area and low-power FIR filter for magnetic recording read channel applications. A parallel TDF architecture operates on real-time input data samples and employs a fast, low-area multiplier based on selection of radix-8 pre-multiplied coefficients in conjunction with one-hot encoded bus leading to a very compact layout and reduced power dissipation. This filter was compared for area, speed and power with other common implementations and it was demonstrated that our approach is most effective for speed critical implementations with the constraints of low cost and low-power. The proposed filter has been fabricated using a 0.18 μm L_{eff} CMOS technology and operates at 550 Msamples/s.

6. REFERENCES

[1] H. Kobayashi and D. Tang, "Application of partial-response channel coding to magnetic recording systems," *IBM J. Res. Develop.*, vol. 14, pp. 386–375, July 1970.

[2] R. Cideciyan *et al.*, "A PRML system for digital magnetic recording," *IEEE J. Select. Areas Commun.*, vol 10, pp. 38–56, Jan. 1992.

[3] K. K. Parhi, "Digital Signal Processing Systems," John Wiley & Sons, Inc. 1999.

[4] L. Thon *et al.*, "A 240 MHz 8-tap programmable FIR filter for disk-drive read channels," *IEEE ISSCC Dig. Tech. Papers*, pp. 82–83, Feb. 1995.

[5] C. Wong *et al.*, "A 50 MHz eigth-tap adaptive equalizer for partial-response channels," *IEEE Journal of Solid State Circuits*, vol. 30, pp. 228–234, Mar. 1995.

[6] H. Ki *et al.*, "A high-speed, low power 8-tap digital FIR filter for PRML disk-drive read channels," *ESSCIRC '97 Conf. Proc.*, pp. 312–315, Sept. 1997.

[7] D. Moloney *et al.*, "Low-power 200-Msps, area-efficient, five-tap programmable FIR filter," *IEEE Journal of Solid State Circuits*, vol. 33, pp. 1134–1138, July 1998.

[8] R. Staszewski and S. Kiriaki, "Top-down simulation methodology of a 500 MHz mixed-signal magnetic recording read channel using standard VHDL," *'99 Behavioral Modeling and Simulation Conf. Proc.*

[9] Texas Instruments Application Specific Integrated Circuits Macro Library Summary, "TSC6000 0.18-μm CMOS Standard Cells," 1998.