International Symposium on Signal Processing and its Applications (ISSPA), Kuala Lumpur, Malaysia, 13 – 16 August, 2001. Organized by the Dept. of Microelectronics and Computer Engineering, UTM, Malaysia and Signal Processing Research Centre, QUT, Australia

DESIGN OF A POWER-SCALABLE DIGITAL LEAST-MEANS-SQUARE ADAPTIVE FILTER

CheeWe Ng, Anantha P. Chandrakasan

MIT Microsystems Technology Laboratory
77 Massachusetts Ave, Room 38-107, Cambridge, MA 02139, USA
cheewe@alum.mit.edu, anantha@mtl.mit.edu

ABSTRACT

Channel equalization is a required function in many high speed communication systems. As a result of the high performance requirements and complexity, adaptive equalization filters require significant power. These filters are often implemented in hardware rather than software on a DSP. In this paper, a power scalable adaptive equalizer is presented where the power scales with the required precision through the use of dynamic tap length and bit precision. Using these techniques, a power reduction from 20.4 mW in a fixed length and bit precision filter to a minimum of 6.4 mW is demonstrated.

1. INTRODUCTION

In recent years, power consumption in digital CMOS has become an increasingly important issue. In portable applications, a major factor in the weight and size of the devices is the amount of batteries. For non-portable devices, cooling issues associated with the power dissipation has caused significant interest in power reduction.

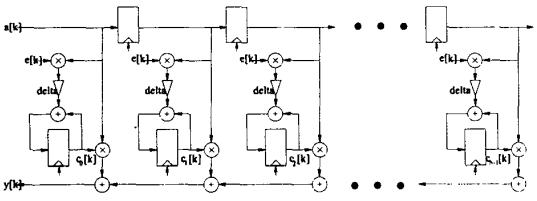

Figure 1: Least Means Square Adaptive Filter.

In this paper, the system design of a power-scalable least means square (LMS) adaptive filter is described. LMS adaptive filters are used for channel equalization in modems and wireless transceivers. Figure 1 shows a block diagram of the filter. $a[k]$ is the input to the filter (received data with inter-symbol interference) at time kT, where T is the symbol period, $c_j[k]$ $(j = 0 \ldots N-1)$ are the N filter taps, $y[k]$ is the output of the filter, and $e[k]$ is the expected output of the filter (received data with the inter-symbol interference removed) minus $y[k]$. The LMS algorithm adjusts the filter taps iteratively to minimize the mean square value of $e[k]$. The reader is referred to references such as [1] for an explanation of the LMS algorithm and its application in channel equalization. For the purposes of this paper, it suffices to note that the algorithm updates the taps as shown in Figure 1, according to the following equation.

$$c_j[k+1] = c_j[k] + \Delta e[k]a[k-j] \qquad (1)$$

where Δ is the adaptation step size.

2. PREVIOUS LOW-POWER DESIGNS

Because of the high rate and computation complexity involved, adaptive equalization filters consume a lot of power. Currently, equalization is typically hardwired instead of using a digital signal processor because current DSPs are unable to handle the large number of operations required per second. [2] cites a requirement of 1440 million operations per second for a decision feedback equalizer.

Previous work on low-power implementation of equalizers include the following. Nicol *et al.* [3, 4] proposed using carry save adders, Wallace-tree multipliers and booth encoding, with adaptive bit precision by adding a programmable gain at the output of the filter. Further power reduction is achieved by having burst-mode update and setting small taps to zero. The delay line (see Figure 1) still needs to operate because other non-zero taps still need to be updated.

Shanbag *et al.* [5] considered shutting off certain taps according to an optimal trade off between power consumption and mean square error. When taps are shut down, the critical path length is reduced, and the supply voltage can be reduced to further reduce power. Additional power savings is achieved by forcing the least significant bits of the input signal and coefficients to zero to reduce their precision, and by using algebraic transformations and pipelining with relaxed lookahead techniques.

In this work, the trade-off between the quality of the cancellation of inter-symbol interference (measured by the

0-7803-6703-0/01/$10.00©2001 IEEE

standard deviation of the steady state output error), and (1) the length of the adaptive filter and (2) the precision of the taps is studied empirically and implemented in CMOS logic to demonstrate power scalability with the required precision.

3. TRADE-OFF BETWEEN STEADY STATE ERROR, AND FILTER LENGTH AND PRECISION

Using 3 channel responses chosen from a set of randomly generated channels with ISI over 2 sample times, the performance of the filter with varying tap length, tap precision and adder precision is studied empirically. Figure 2 shows plots the standard deviation of the error as the tap length, tap precision and adder precision are varied.

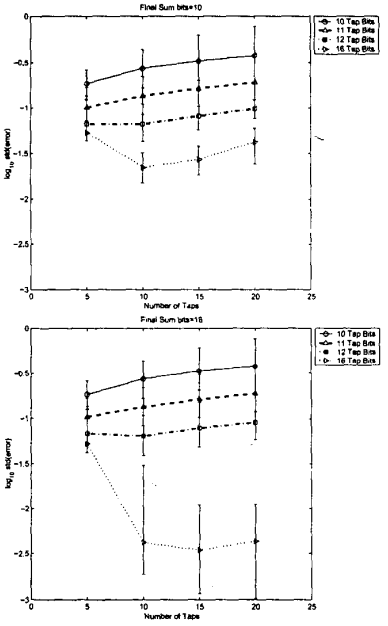

Figure 2: Performance of filter with varying tap length, tap precision and adder precision for ISI over two time samples.

Referring to Figure 2, the following observations are made. First, given the number of taps and the precision of the final sum, a higher tap precision leads to a smaller error and better performance. Second, a larger number of taps may lead to poorer performance, especially with low precision taps (10,11 and 12). As discussed in [6] the minimum filter order needed for a certain error rate is not well understood. The reader is referred to [1] for discussions on the effect of tap length and precision on the mean square error of the output. Finally, comparing the two graphs in

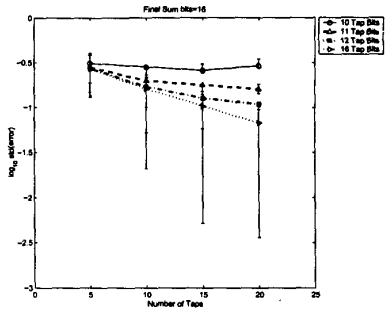

Figure 3: Performance of filter with varying tap length, and tap precision for ISI over three time samples.

Figure 2, the precision of the final sum only improves the performance of the filter when the taps have high-precision (in this case 16 bits). With an ISI that covers three time samples (see Figure 3), a larger number of taps led to better performance most of the time.

4. A POWER-SCALABLE IMPLEMENTATION

Figure 4 shows the overall architecture of an implementation of a fifteen tap scalable adaptive filter. The design in this work has four levels of adjustability: (1) Five 11-bit taps, (2) Ten 11-bit taps, (3) Ten 16-bit taps, and (4) Fifteen 16-bit taps. For simplicity, the precision of the delay line is kept constant, although reducing it dynamically would have given us greater power dissipation reduction.

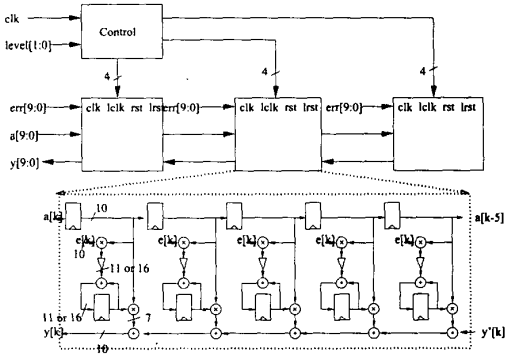

Figure 4: Overall architecture.

The filter consists of three sub-blocks controlled by a control block. All the signals to each sub-block are local to the sub-block and generated by the control block. The signals' respective functions are summarized in Table 1 and will be explained in the following subsections.

Table 1: Control signals for each sub-block. When a sub-block is turned off, all clock signals are gated, and resets are set high.

Signal	Function
clk	clock for the block except the least-significant tap bits
$lclk$	clock for the least-significant tap bits
rst	asynchronously resets all registers except the least-significant tap bits
$lrst$	asynchronously resets all the least-significant tap bits activates low-precision mode for the sub-block

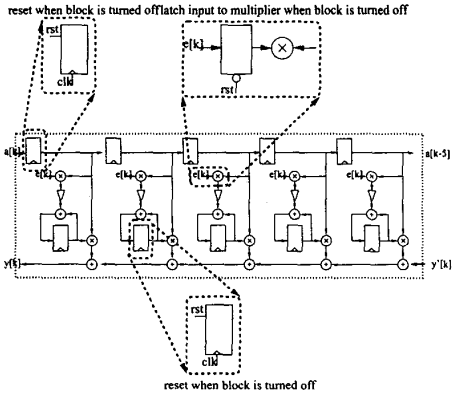

Figure 5: Turning off a block by gating the clocks, reseting the registers, and latching the input to multipliers.

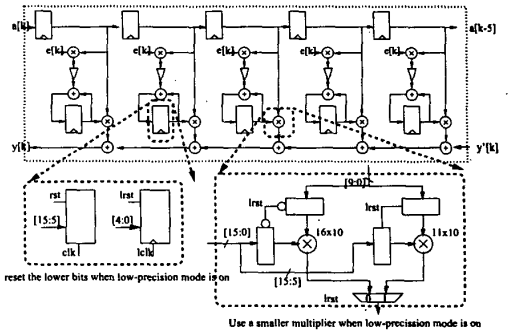

Figure 6: Low-precision mode by gating the clock to the lower precision bit registers and using lower precision multipliers.

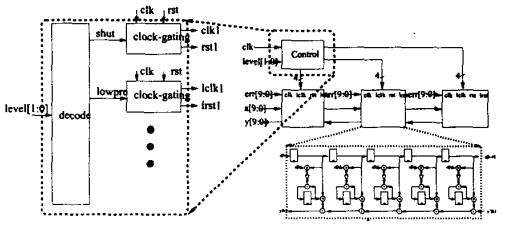

Figure 7: Architecture of control logic.

4.1. Implementing Adjustable Length

To shut off the latter two filter blocks when not needed, the clk signals of these blocks are gated. In addition, the inputs to the tap-update multiplier are latched, since the $err[9:0]$ is still changing. All the registers are also asynchronously reset so that when these blocks are restarted, they do not contain any old values. The latching and asynchronous reset are controlled by the rst signal of each sub-block as shown in Figure 5.

4.2. Implementing Adjustable Tap-precision

Adjustable tap-precision is implemented as follows. The clock $lclk$ of the lower order bits of the tap registers are gated when operating in low-precision, as shown in Figure 6. A simple bit-length adjustable multiplier shown in Figure 6 is used to reduce power consumption further. It consists of two multipliers of different input bit-length. When a higher precision is needed, the mux selects the output from the longer bit-length multiplier and the inputs to the other are latched so they are not affected by the changing inputs. Note how this approach differs from both Nicol *et al.* [3, 4], who used a programmable gain at the output of the filter, and Shanbag *et al.* [5], who simply set the least significant bits to zero. Using a shorter bit-length multiplier instead of a longer bit-length multiplier reduces power dissipation when the precision is not required. In 0.18μm technology, the area of duplicating the multiplier is relatively small though the multipliers could be further optimized for power consumption.

4.3. Control Logic

The control logic performs two functions. First, it decodes the $level[1:0]$ signal into a signal for whether each block should be turn on or off ($shut$) and whether each block should be running high tap precision mode or low tap precision mode ($lowpre$). Second, it generates the clk, $lclk$, rst, and $lrst$ signals for each sub-block. This is summarized in Figure 7.

The first portion is a simple combinational logic that computes the $shut$ and $lowpre$ signal for each sub-block. The second portion is slightly more elaborate because we allow the $level[1:0]$ signal to be asynchronous. When $shut$ is low, a inverted clk and rst signal for a filter sub-block is derived from the clk, and rst signal. When $shut$ goes high, derived clock is held high at the subsequent rising clk edge. It is held high until the first rising clk edge following the falling edge of $shut$. During the time the de-

rived clock is held high, rst of the sub-blocks are held high to reset the internal registers and to latch the inputs to the tap-update multiplier. The implementation of this block is shown in Figure 8.

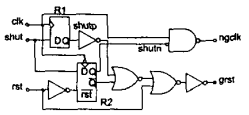

Figure 8: Implementation of clock-gating circuit.

5. IMPLEMENTATION RESULTS

The final layout in Figure 9 has a transistor count of 117,315 and dimensions of 743μm by 773μm in 0.18μm technology, excluding the pads. When clocked at 33 MHz with a power supply of 1.8 volts, the power consumption is summarized in Tables 2 and 3.

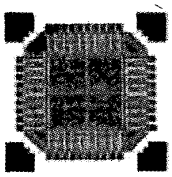

Figure 9: Layout of circuit using TSMC 0.18 μm Process.

Table 2: Breakdown of power dissipation (mW), clocked at 33 MHz with a power supply of 1.8 V.

Element	Level 0 5 11-bit Taps	Level 1 10 11-bit Taps	Level 2 10 16-bit Taps	Level 3 15 16-bit Taps
Clock	0.06 (0.9%)	0.13 (1.1%)	0.19 (1.3%)	0.27 (1.3%)
Delay line registers	0.12 (1.9%)	0.26 (2.3%)	0.26 (1.8%)	0.40 (1.8%)
Tap registers	0.12 (1.9%)	0.23 (2.2%)	0.35 (2.3%)	0.53 (2.3%)
Update multiplier	2.25 (35.0%)	3.85 (34.0%)	3.45 (23.0%)	4.21 (18.4%)
Tap update adder	0.50 (7.7%)	0.94 (8.3%)	1.34 (8.9%)	1.86 (8.1%)
Tap multiplier	2.51 (38.3%)	3.97 (35.0%)	7.17 (47.7%)	9.99 (43.6%)
Final sum	0.67 (10.4%)	1.46 (12.9%)	1.64 (11.0%)	2.69 (11.7%)
Total	6.45	11.33	15.00	22.92

Using the clock-gating technique, we obtain power savings in the clock network, the delay line and tap registers, multipliers and adders in the filter sub-blocks that are turned off. Using adaptive precision for the taps, we are also able to scale down the power dissipation of the tap multiplier and tap registers linearly with the tap precision. Figure 10 shows the trade-off between power consumption and standard deviation of error of the filter.

6. CONCLUSIONS

In this paper, a trade-off between power dissipation and the quality of ISI-cancellation of an LMS adaptive filter has been demonstrated. The circuit designed demonstrated a power scalability from 6.4 to 20.4 mW with the corresponding output quality measured by standard deviation of error ranging typically from 0.28 to 0.1. Our simulations show that shorter filter length and smaller tap precision lead to larger standard deviation of error, but also consume less power.

Table 3: Breakdown of switched capacitances(pF).

Element	Level 0 5 11-bit Taps	Level 1 10 11-bit Taps	Level 2 10 16-bit Taps	Level 3 15 16-bit Taps
Clock	0.6	1.2	1.7	2.5
Delay line registers	1.1	2.4	2.5	3.8
Tap registers	1.1	2.3	3.2	4.9
Update multiplier	20.9	47.2	31.9	39.0
Tap update adder	4.6	8.7	12.3	17.2
Tap multiplier	23.2	36.7	50.2	92.5
Final sum	6.2	13.5	15.2	24.8
Total	59.7	104.9	138.8	192.0

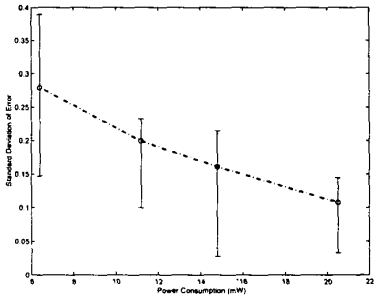

Figure 10: Trade-off between power and standard deviation of error at output.

7. REFERENCES

[1] S.U.H Qureshi, "Adaptive Equalization," in *Proceedings IEEE*, September 1985, vol. 73, pp. 1349–1387.

[2] Fang Lu and Henry Samueli, "A 60-Mbd, 480-Mb/s, 256-QAM Decision-Feedback Equalizer in 1.2-μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 3, pp. 330–338, March 1993.

[3] Chris J. Nicol, Patrik Larsson, Kamran Azadet, and Jay H. O'Neill, "A Low-Power 128-Tap Digital Adaptive Equalizer for Broadband Modems," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1777–1789, November 1997.

[4] Kamran Azadet and Chris J. Nicole, "Low-Power Equalizer Architectures for High-Speed Modems," *IEEE Communications Magazine*, pp. 118–126, October 1998.

[5] M. Goel and N.R. Shanbag, "Low-Power Equalizers for 51.84 Mb/s Very-High-Speed Digital Subscriber Loop (VDSL) Modems," in *1998 IEEE Workshop Signal Processing Systems*, October 1998, pp. 317–326.

[6] J.R. Treichler, I. Fijalkov, and C.R. Johnson Jr., "Fractionally Spaced Equalizers," *IEEE Signal Processing Magazine*, May 1996.