Power Reducing Algorithms in FIR Filters

by

Nitin Kasturi

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1997

Author .....
Department of Electrical Engineering and Computer Science
May 15, 1997

Certified by .....
Anantha Chandrakasan
Analog Devices Career Development Assistant Professor of Electrical Engineering
Thesis Supervisor

Certified by .....
Alan V. Oppenheim
Ford Professor of Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Power Reducing Algorithms in FIR Filters

by

Nitin Kasturi

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 1997, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering

Abstract

Power consumption has become a major concern in the design of integrated circuits for reasons of portability, miniaturization, and conservation. Accordingly, there has been a need to find ways to reduce power. One fundamental area of interest is the multiplication component in the convolution process of a Finite Impulse Response (FIR) filter. Two classes of power reduction techniques have been developed and investigated. Both classes approach the problem of power with an intent to reduce transition activity. Transition activity is an important factor in switching power, a major component of power consumption.

The first class uses block processing and additional memory to reorder the multiplications inherent in the convolution process. Two such methods are explored. The first seeks to maintain the coefficient input to the multiplier as constant for several multiplications. The second method, instead, maintains the signal input to the multiplier. Through simulations across a range of multipliers, it was found that the first method (termed vertical unrolling) was found to reduce power more.

The second class of power reduction techniques involved the perturbation of the coefficients of the filter while maintaining various filter design constraints. In the developed algorithm, the coefficients are first scaled within an allowable gain and then perturbed though an iterative process that sets low-end bits of sets or subsets of the coefficients equal. Through this method, coefficient transition activity was reduced up to 56%. Power in the multiplier was reduced in the best case by about 25%. This approach does not need any additional memory requirements. The cost of the power reduction is a slight alteration of the results of the convolution.

Thesis Supervisor: Anantha Chandrakasan
Title: Analog Devices Career Development Assistant Professor of Electrical Engineering

Thesis Supervisor: Alan V. Oppenheim
Title: Ford Professor of Engineering

Acknowledgments

This thesis represents the conclusion of five wonderful years at MIT. I would like to take this opportunity to thank all the professors, TAs, and fellow students who made this such a great rewarding experience.

For this thesis, I would like to acknowledge several people who helped make it happen. First, I'd like to mention my fellow graduate students in Course VI who supported me, while at the same time were going through the same process themselves – Brad Bartley, Erika Chuang, Joel Dawson, Scott MacGregor, Jennifer Shen, and Julianne Zhu.

While researching and writing this thesis, several graduate students selflessly provided valuable assistance – Abram Dancy, Jim Goodman, Jeff Ludwig, Wendi Rabiner, and Thucydides (Duke) Xanthopoulos.

This thesis would definitely not have happened without the guidance and support of my thesis advisors – Anantha Chandrakasan and Al Oppenheim. They helped me develop several ideas and made sure I thoroughly investigated them.

Finally, but foremost, I would like to thank my parents – Srinivasan and Tara Kasturi – for allowing me to go to MIT and making sure I did my best while I was here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Increasingly, the reduction of power consumption in digital systems has become a major design issue. This consideration is driven by two primary forces – battery operated portable computing and heat dissipation in high performance processors.

There has been a significant increase in the use of portable battery operated devices such as cellular phones, laptop computers, and PDAs. Users desire lightweight devices that are capable of running for extended periods of time. Thus, these devices must be able to make optimal use of battery power. By reducing power consumption, batteries on such devices may then be made lighter, smaller, and longer-lasting.

Even in non-portable applications, power issues are still a concern. The scaling of feature sites to the submicron regime has resulted in millions of transistors being integrated on a single chip. The scaling has closely followed Moore's Law, which predicts that the number of transistors per chip quadruples every three years. As integrated circuits get denser, the ability of cooling fans and other external heat dissipation sinks to quickly remove excess heat becomes severely limited. By reducing the power that is consumed by devices, designers can remove some of the restrictions placed on them by cooling requirements.

The third driving force behind power consumption has been in the interests of conservation. Personal computers in the U.S. have been shown to use $ 2 billion dollars of electricity, indirectly produce as much CO_2 as 5 million cars, and account for 5% of commerical electricity consumption. These figures provide motivation to incorporate power-reducing technology into digital systems. [3]

Current systems are often built around embedded processors, one important type being

Digital Signal Processors (DSPs). One of the most common components of DSP systems is a Finite Impulse Response (FIR) filter. Filtering by an N-tap FIR filter is equivalent to the convolution of a finite-length time-domain filter (N coefficients) with a discrete-time input signal, and is represented by the equation,

$$y_k = \sum_{i=0}^{N-1} c_i x_{k-i} = c_0 x_k + c_1 x_{k-1} + \ldots + c_{N-1} x_{k-(N-1)}, \tag{1.1}$$

where the c_i's are the coefficients of the FIR filter, and the x_k's and y_k's are the discrete-time input and output signals, respectively.

All physical representations of the convolution of an FIR filter involve some sort of multiply and accumulate process, as suggested by Equation 1.1. In an FIR filter, multiplications have been shown to contribute significantly to the total power consumed during filtering. In several types of multipliers, maintaining the inputs to the multiplier and, in general, reducing the transitions between successive inputs will reduce power consumption significantly. Consequently, careful ordering of the multiplications involved in the convolution process will allow multiplier power consumption to be reduced. Furthermore, these ideas can be generalized to special cases of matrix multiplication. Any inherent symmetries within the matrices can be used to reorder multiplications and reduce power consumption. Additionally, the dependence of power in the multiplier on transitions in the inputs motivates another approach in the case of convolution. The coefficients of the filter can be perturbed slightly without significant change to the frequency response of the filter. Within the constraints of maintaining this frequency response, the perturbation can reduce the number of transitions between successive coefficients. It will be seen that an algorithm that effects this purpose will be able to reduce the power consumption of the multiplier.

Chapter 2

Power Consumption in CMOS Devices

2.1 Sources of Power

Although peak power consumption is an important design issue, average power is more critical in considering the long-term lifetimes, sizes and weights of batteries. Average power consumption in digital CMOS circuits can be tied to four main sources. These are, for a given CMOS gate,

$$P_{avg} = P_{switching} + P_{short-circuit} + P_{leakage} + P_{static} \quad (2.1)$$

$$= \alpha_{0\to1} C_L V_{dd}^2 f_{clk} + I_{leakage} V_{dd} + I_{sc} V_{dd} + I_{static} V_{dd}. \quad (2.2)$$

$P_{switching}$ is the switching (or dynamic) component of power, where $\alpha_{0\to1}$ is the switching activity factor (the probability of a $0 \to 1$ transition during a clock cycle, or, equivalently, the average number of transitions at a node during a clock cycle), C_L is the load capacitance of the gate, V_{dd} is the supply voltage, and f_{clk} is the system clock frequency. The second component, short-circuit power, $P_{short-circuit}$, is attributable to the direct-path short-circuit current, I_{sc}. This condition occurs when both the NMOS and PMOS transistors of a gate are active, allowing current to flow from supply to ground. The next component, leakage power, $P_{leakage}$, is caused by leakage current, $I_{leakage}$, which results from reverse bias diode currents and sub-threshold effects, which arise in the course of fabrication. Finally, static power, P_{static}, is a result of static currents, I_{static}, that arise from circuits that have a

constant source of current between the power supplies. [3]

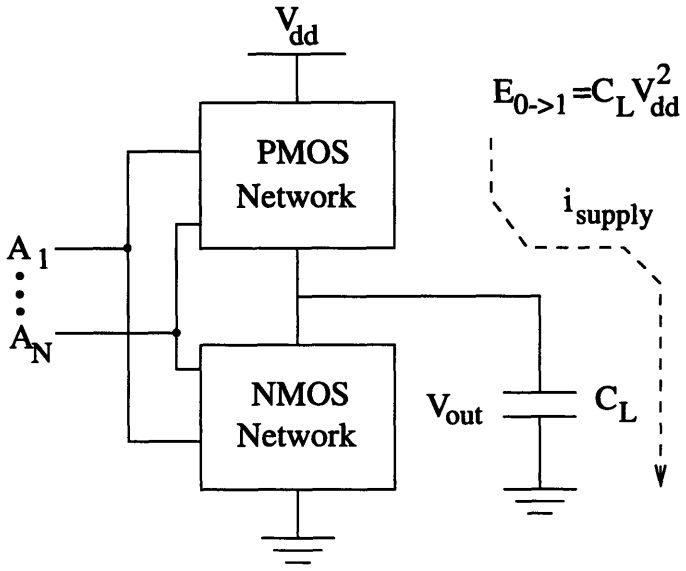

Figure 2-1: Circuit diagram of CMOS gate and power drawn from power supply in a $0 \rightarrow 1$ transition.

2.2 Switching Power

Among the terms in Equation 2.1, more than 90% of power is attributed to the switching component. Therefore, this thesis will focus on the minimization of switching power. Figure 2-1 shows a CMOS gate with the physical load capacitance explicitly included. Switching power arises when the CMOS gate undergoes a transition from $0 \rightarrow 1 (V_{dd})$ or from $1 \rightarrow 0$. Energy is drawn from the power supply for a $0 \rightarrow 1$ transition at the output. Ignoring short-circuit currents, the instantaneous power is given by

$$P(t) = \frac{dE}{dt} = i_{supply} V_{dd}, \tag{2.3}$$

where i_{supply} is the instantaneous current drawn from the power supply through the capacitative load:

$$i_{supply} = C_L \frac{dV_{out}}{dt}. \tag{2.4}$$

Thus, the energy drawn from the power supply during the $0 \rightarrow 1$ transition is given by

$$E_{0\rightarrow 1} = \int_0^T P(t)dt = V_{dd}\int_0^T i_{supply}(t)dt = V_{dd}\int_0^{V_{dd}} C_L dV_{out} = C_L V_{dd}^2, \qquad (2.5)$$

where T is the duration of the transition. The integration was changed from one in time to one in output voltage using Equation 2.4 and by noting that at the end of the transition the capacitor is fully charged to V_{dd}. From this derivation, it is seen that the energy drawn from the power supply in a $0 \rightarrow 1$ transition is independent of the shape of the output voltage waveform.

The average energy of $0 \rightarrow 1$ transitions is obtained by multiplying $E_{0\rightarrow 1}$ by the probability of any such transition on a given clock cycle, α. The average switching power drawn from the power supply is thus this energy times the clock frequency,

$$P_{switching} = \alpha_{0\rightarrow 1} E_{0\rightarrow 1} f_{clk} = \alpha_{0\rightarrow 1} C_L V_{dd}^2 f_{clk} \qquad (2.6)$$

2.3 Power Simulation using PYTHIA

Most of the power estimates presented in this thesis were simulated by PYTHIA, a power estimation tool that works with a structural Verilog circuit description. PYTHIA operates at the gate/module level rather than at the transistor level. In PYTHIA, every circuit net has an associated capacitance value. As the simulation involving a circuit progresses, any transitions between 0 to 1 or 1 to 0 result in energy being dissipated on the order of $\frac{1}{2}CV^2$. Upon completion of the simulation, the total dissipated energy is divided by the total simulation time to obtain the average power dissipated during the simulation. One limitation of PYTHIA is that it is only capable of measuring the switching power contribution in Equation 2.1. However, as stated in the last section, most of the power consumption is attributable to this switching power component and, therefore, this limitation will not be a significant concern.[11] All power simulations in this thesis were performed using the hp14 technology set with a voltage of 3.3 volts. One multiplication was computed every 40 nanoseconds.

2.4 General Methods to Reduce Switching Power

There have been many approaches to reducing this switching power component. These methods can be categorized in four broad categories: (1) technology level, (2) circuit level, (3) architecture level, and (4) algorithm level.

One architecture level method of reducing switching power makes use of its dependence on the clock frequency and the supply voltage. If, at any point while a circuit is in use, either of these variables are zero, no power will be consumed, switching or otherwise. Thus, when no computation is necessary, the voltage applied may be turned off, and no power is consumed. This method is limited to cases for which there are gaps in operation where the circuit is not needed and may be turned off. However, this approach has limited benefits and is useless when the circuit is actually on.

One method that combines a circuit level approach with an architectural one once again takes advantage of the dependence of switching power on supply voltage and clock frequency. By using parallel computation blocks, the clock rate can be reduced without loss in throughput. For example, N-way parallelism allows the clock rate to be dropped by a factor of N while preserving the original throughput. Hence the supply voltage can be dropped without loss in performance. With the supply voltage and the clock frequency decreased, the switching power is consequently decreased. [3]

Many algorithm-level methods reduce switching power by decreasing the activity factor in Equation 2.6. For example, in an array multiplier, power dissipation is directly proportional to the number of switchings in all the internal nodes of the multiplier. These switchings depend on the multiplier input values and transitions. The transition density, the average number of transitions of the signal per unit time, is measured by the Hamming distance. The Hamming distance for a signal is the number of total bits that toggle between successive samples of the signal. Many algorithm-level methods seek to decrease power consumption by decreasing the Hamming distance. [6]

All of the power-reducing methods explored in this thesis seek to reduce power consumption by using algorithm-level methods that reduce the Hamming distance.

Chapter 3

Test Data

3.1 FIR filter

In order to test the validity of power-reducing algorithms using the FIR filter computation of Equation 1.1, two data streams were needed. One was c_i, the FIR filter with length N. In all of the FIR filter problems, a Hamming window of length 16 was used as the FIR filter. This filter can be seen in Figure 3-1. The filter was scaled to fully span a 16-bit sign-magnitude form.

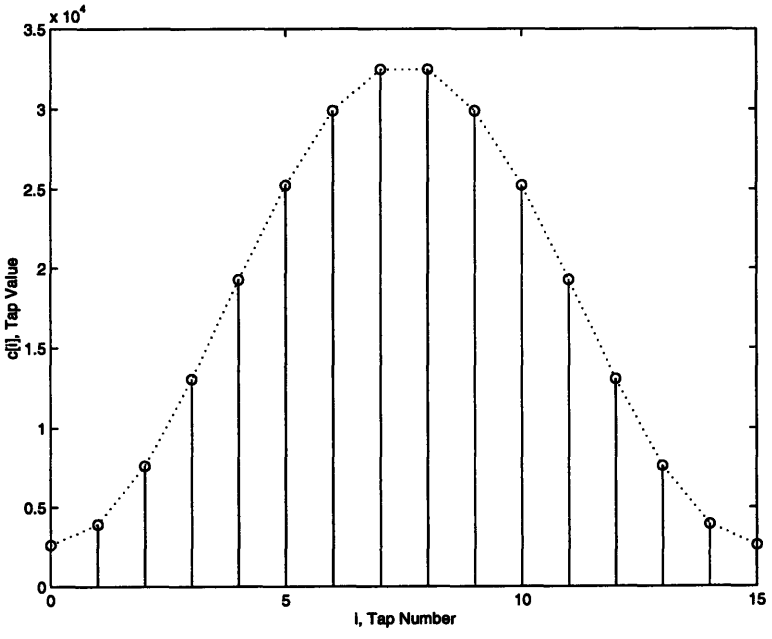

Figure 3-1: A 16-tap Hamming Window

3.2 FIR Signal Data

The second data stream that was needed was an input signal, x_i. This signal was synthesized with input parameters detailing the length (N), mean (μ), standard deviation (σ), and correlation (ρ) of the data. For use with the different multipliers, the same data set was represented in binary in both two's complement and sign-magnitude forms. It was also necessary to provide a seed for the random number generator used by the program. The data would be generated according to the recursion,

$$x_0 = \mu + \sigma\mathcal{R} \tag{3.1}$$

$$x_i = \rho x_{i-1} + \sigma\sqrt{1-\rho^2}\mathcal{R}, \qquad \text{for } i = 1 \ldots N, \tag{3.2}$$

where $\mathcal{R}$ is a zero-mean, unit-variance normally distributed random number. The first term of x_i represents the part that is correlated with the previous sample. The second term is randomly generated to be uncorrelated with the previous sample. The multiplicative terms preserve the energy of the signal. In practice, though, correlation is also a random variable. The expected value of this correlation random variable will be the correlation that is used to generate the stream. However, since there is a variance associated with this variable, the measured correlation of a data stream will not equal the correlation used for generating the stream.

Two sets of data were created, one with a standard deviation of 1000 and another with a standard deviation of 100. Each set contained data files ranging in correlation from -0.9 to 0.9, in increments of .1. Table 3.1 shows the correlations used to create the data and the measured correlations.

Correlation was deemed an important parameter because data that was more highly correlated would tend to have more bits in common, thus having a lower transition activity[5]. In order to confirm this, the transition activity of each data file was analyzed bit-by-bit. The results are displayed in Figure 3.2. The horizontal axis of each graph runs from most significant bit (MSB) at 1 to least significant bit (LSB) at 16. The vertical axis displays the probability of a zero to one transition ($P(0 \rightarrow 1)$). The importance of this quantity to switching power is readily apparent as this probability is precisely the factor, $\alpha_{0\rightarrow 1}$, in Equation 2.6. A purely random $0 \rightarrow 1$ transition would have a probability of .25. (There are four possible transitions – $0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0$, and $1 \rightarrow 1$.) For each graph

$\sigma = 1000$				$\sigma = 100$			
ρ_{synth}	ρ_{actual}	ρ_{synth}	ρ_{actual}	ρ_{synth}	ρ_{actual}	ρ_{synth}	ρ_{actual}
-.9	-.9014	.1	.0931	-.9	-.8884	.1	.1008
-.8	-.8184	.2	.2154	-.8	-.8009	.2	.2461
-.7	-.7061	.3	.2633	-.7	-.6698	.3	.3148
-.6	-.6036	.4	.3650	-.6	-.6263	.4	.3221
-.5	-.4970	.5	.5059	-.5	-.5113	.5	.4986
-.4	-.3983	.6	.5740	-.4	-.4218	.6	.6359
-.3	-.2512	.7	.6940	-.3	-.3488	.7	.6805
-.2	-.1937	.8	.7580	-.2	-.1723	.8	.7816
-.1	-.1002	.9	.8729	-.1	-.1205	.9	.8962
0	-.0251			0	-.0318		

Table 3.1: Actual correlation values for data generated with ρ_{synth}

the transitions of the whole set of data in the appropriate form with the given standard deviation, σ, is shown. Only the transition lines of the extreme cases, $\rho = 0.9$ or $\rho = -0.9$, are labeled as the transitions lines proceed sequentially between these extremes.

From the graphs of the sign-magnitude cases, it can be seen that for the most part correlation has little effect on transition activity. The only exception is the MSB, or the sign-bit. However, it will be observed that in a sign-magnitude multiplier the sign-bit is used independently of the lower bits in simple combinational logic to compute the sign-bit of the result. Accordingly, it can be reasoned that the sign-bit contributes little to the total power consumption. Another point that can be observed is that the bit-value at which the probabilities of a $0 \rightarrow 1$ transition are random (.25) is lower (starts at a less significant bit) for data with a lower standard deviation. This agrees with intuition as data with a lower standard deviation is more closely distributed about zero.

Looking at the two's complement form, it is apparent that correlation has a greater impact on the transition activity of data. The high-bits of more highly-correlated data are less likely to transit than less correlated or negatively correlated data. Furthermore, as in the sign-magnitude case, the data assumes a random nature at lower bits. In the two's complement case, the breakpoints of the transition activity have been quantified in terms of relevant parameters.[5] Starting from the MSB, the probability maintains its flat structure, indicative of the sign of the number, until it reaches some first breakpoint, BP_1. The probability of transition then changes to a random nature at a second breakpoint, BP_0.

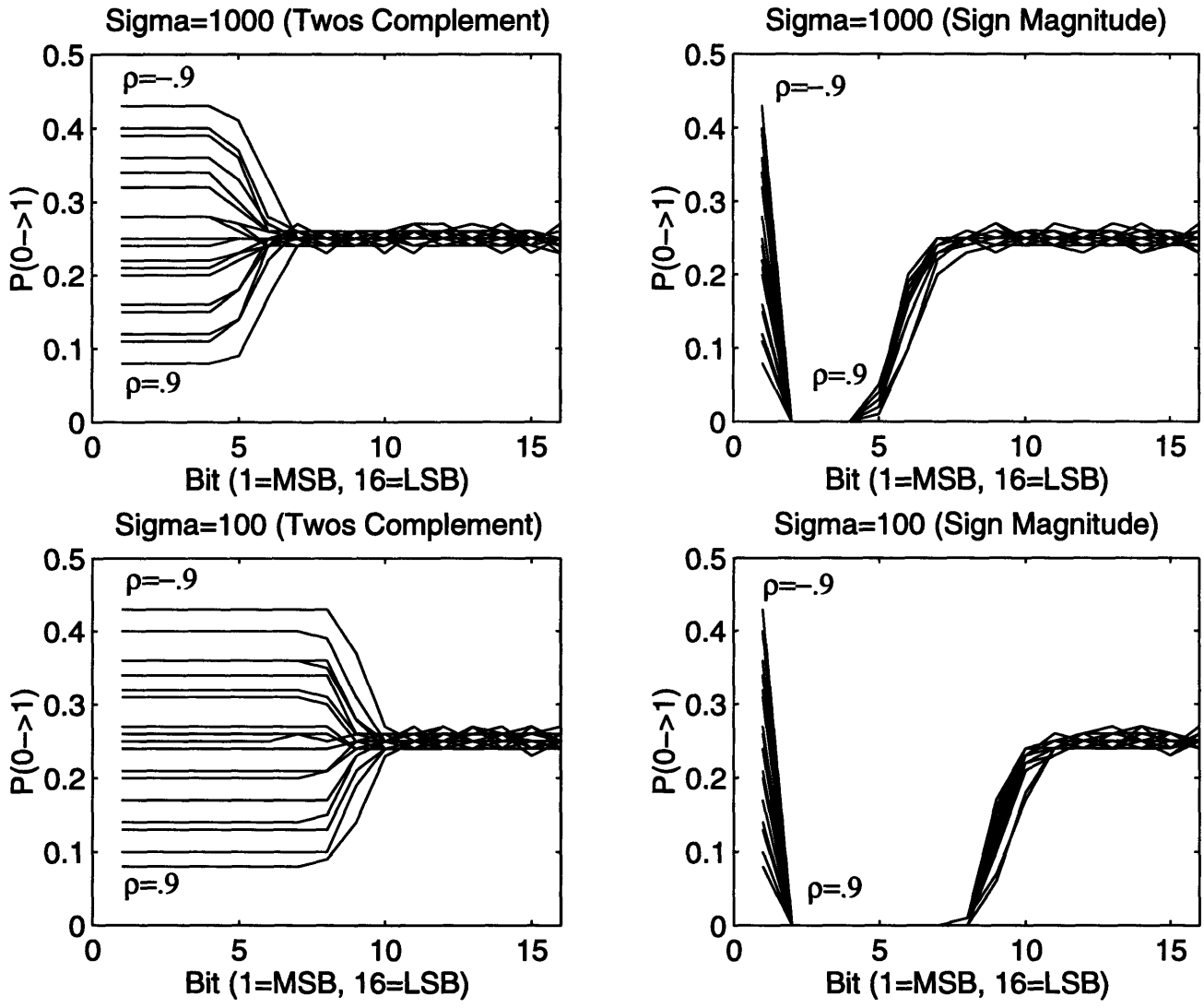

Figure 3-2: Transition activity for different representations of two sets of data.

These breakpoints are

$$BP_1 = N - \log_2(|\mu| + 3\sigma) \quad (3.3)$$

$$BP_0 = N - (\log_2 \sigma + \Delta BP_0) \quad (3.4)$$

$$\Delta BP_0 = \log_2(\sqrt{1-\rho^2} + \frac{|\rho|}{8}), \quad (3.5)$$

where N is the bit-width of the data. The breakpoints for selected data used are calculated in Table 3.2. For two's complement data with $\sigma = 1000$, the range of the transition from the end of the flat, sign-bit portion of the number (BP_1) to the start of the random bit transitions (BP_0) is from bit 4 to about bit 7. This agrees with the upper left graph of Figure 3.2. For two's complement data with $\sigma = 100$, the range of the transition is from bit 6 to about bit 10, which again conforms with the lower-left graph in the figure.

Breakpoint	$\mu = 0$ $\sigma = 1000$ $\rho = \pm.9$	$\mu = 0$ $\sigma = 1000$ $\rho = 0$	$\mu = 0$ $\sigma = 100$ $\rho = \pm.9$	$\mu = 0$ $\sigma = 100$ $\rho = 0$
BP_1	4.45	4.45	7.77	7.77
BP_0	6.90	6.03	10.22	9.36
ΔBP_0	-.867	0	-.867	0

Table 3.2: Transition breakpoints for selected synthesized data sets in terms of bits (1=MSB, 16=LSB).

3.3 Toeplitz Data

The data used to test out multiplications involved in a Toeplitz matrix multiplication was derived from an actual speech waveform. The symmetries of a Toeplitz matrix and the actual components within the matrix multiplication shall be discussed in Chapter 4.3.

3.4 Multipliers

The data described in this chapter was used in simulations of multiplications to estimate the power consumed in various routines. Three distinct multipliers were used in these simulations. The first one was a sign-magnitude array multiplier. The other two multipliers both took data in a two's complement form. The Baugh-Wooley multiplier was based

structurally on the sign-magnitude array multiplier. The Booth-Encoding multiplier used several bits of an input at a time to sequentially compute the result. A complete description of the multipliers may be found in Appendix A. In summary, the conclusions drawn about these multipliers are as follows. In general, the Baugh-Wooley multiplier will consume more power than the sign-magnitude array multiplier, since, in the former multiplier, the sign of the inputs is implicit in the number and an additional array level is necessary. Also, it is concluded that the Booth-Encoding multiplier will not exhibit a significant dependence on the correlation of its data inputs since several bits of an input are used at a time, and all of these bits must be identical for the multiplication elements to remain the same. Therefore, it is believed that the sign-magnitude multiplier will consume the least power.

The motivation behind all of the routines considered in this thesis is that the power consumed in a multiplier is dependent on the number of transitions at its inputs. This statement was validated by multiplying data streams by a constant using a sign-magnitude array multiplier. The explicit dependence of power on the transition probability can be seen in the results plotted in Figure 3-3.

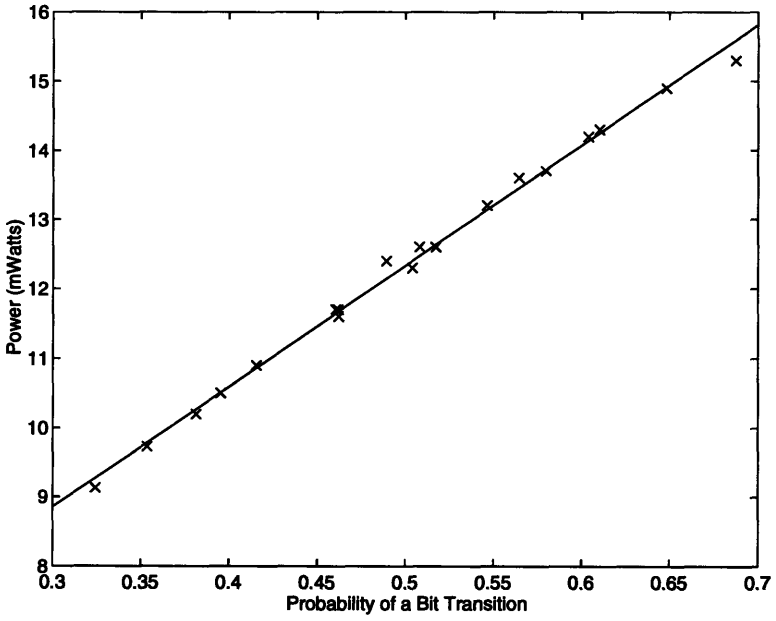

Figure 3-3: The dependence of power on the probability of non-sign-bit transitions

Chapter 4

Unrolling

4.1 Vertical Unrolling

4.1.1 Concept

One of the methods that we considered to reduce transition activity used block processing to compute the outputs of the convolution of an input signal with an FIR filter. Block processing involves computing several outputs of the convolution simultaneously rather than sequentially. The conventional order of multiplication within the convolution, as denoted by the bold superscript numbers, is

$$y_0 = c_0\overset{\mathbf{1}}{x}_0 + c_1\overset{\mathbf{2}}{x}_{-1} + c_2\overset{\mathbf{3}}{x}_{-2} \tag{4.1}$$

$$y_1 = c_0\overset{\mathbf{4}}{x}_1 + c_1\overset{\mathbf{5}}{x}_0 + c_2\overset{\mathbf{6}}{x}_{-1}, \tag{4.2}$$

where the c_i's are the coefficients of the FIR filter, and the x_i's and y_i's are the input and output signals, respectively.

In vertical unrolling, block processing is used, and the coefficients, or taps are kept constant for as many multiplications as possible. This will lower power consumption within the multiplier. The argument for this is based on Equation 2.1. Switching power is directly dependent on the probability of transition, $\alpha_{0\to1}$. By rearranging the multiplications involved in the convolution, the number of transitions in the signal inputs to the multiplier can be decreased. In the case of vertical unrolling, keeping inputs constant over successive multiplications results in no transitions in those inputs for those multiplications. Based on

the formula for switching power, this should reduce power consumption.

For an unrolling of order 3, the order of multiplications is

$$y_0 = \overset{1}{c_0 x_0} + \overset{6}{c_1 x_{-1}} + \overset{7}{c_2 x_{-2}} \tag{4.3}$$

$$y_1 = \overset{2}{c_0 x_1} + \overset{5}{c_1 x_0} + \overset{8}{c_2 x_{-1}} \tag{4.4}$$

$$y_2 = \overset{3}{c_0 x_2} + \overset{4}{c_1 x_1} + \overset{9}{c_2 x_0}\,. \tag{4.5}$$

One thing that can be noted is that not only are the taps held constant but also that the signal always switches by only a single sample when the multiplications are ordered in this manner (as opposed to all multiplications performed vertically downward). Thus, as seen in Section 3.2, data in some representations has a high first-order correlation and exhibits less transitions than purely random data. This will have the added benefit of reducing transitions in the other input to the multiplier.

4.1.2 Results

The set of tables in this subsection show the results of vertical unrolling with various data sets and types of multipliers. Block processing of order 1, 2, 3, and 4 was performed with a sign-magnitude array multiplier, a Booth-Encoding two's complement multiplier, and a Baugh-Wooley two's complement multiplier. The two data sets, described in Section 3.2, have standard deviations of 100 and 1000 and contain data signals ranging in correlation from -0.9 to 0.9. The range of the power reduction using vertical unrolling relative to the conventional method of computing the convolution without any unrolling can be found in Table 4.7.

First, looking at the sign-magnitude array multiplication data in Tables 4.1 and 4.4, as expected there is no variation in the power results with respect to correlation. This was expected because of the behavior of the transition probability of the sign-magnitude relative to correlation. Looking at the right two graphs of Figure 3.2, it is seen that the transition activity of the sign-magnitude data is largely independent of correlation. Only the most significant bit (MSB), the sign-bit, has any dependence on correlation. Yet, as seen in Appendix A, the combinational logic for the sign-bit of the array multiplier is quite simple and does not consume a significant amount of power. Therefore, no dependence on correlation is seen in the power simulations.

The Booth-Encoding multiplier was largely independent of the correlation of the data as well, as seen in Tables 4.2 and 4.5. As explained in more detail in Appendix A, the Booth-Encoding multiplier observes several bits of an input at a time and adds or subtracts the other input based on those bits. Thus, the state of the circuit for those few bits will be identical in successive multiplications only if all of the few bits are exactly the same in the consecutive inputs. Even though the transition probabilities of two's complement vary with correlation, as seen in the left two graphs of Figure 3.2, there is no reason that the variations of identical bits will be the same between successive data inputs. Thus, correlation will not affect power consumption significantly; however, in the extreme case when the inputs are held constant, power will be reduced since all of the subset bits remain the same.

The Baugh-Wooley multiplier was significantly affected by the correlation of the data. As seen in Section 3.2, correlation affects the transition probabilities of higher-order bits when the data is in two's complement form. Negatively correlated data transitions more than positively correlated data; consequently, since power consumption is proportional to the number of transitions, in Tables 4.3 and 4.6, we see that negatively correlated data consumes more power. Furthermore, as seen in Section 3.2, the transition breakpoints are lower (less significant bit) for data with a lower standard deviation. Thus, looking at Figure 3.2, for positively correlated data, more bits exhibit lower transition probabilities for $\sigma = 100$ data. Therefore, we see a greater dependence of power on correlation in the $\sigma = 100$ data set.

Comparing the three multipliers, we see that the sign-magnitude multiplier consumes the least power. This was expected especially relative to the Baugh-Wooley multiplier since this multiplier required additional dimensions within its array than the sign-magnitude array multiplier. The Baugh-Wooley and Booth-Encoding multipliers are more on par in power requirements, though the Baugh-Wooley multiplier is apt to consume less power when data is positively correlated.

As summarized in Table 4.7, the power reduction through vertical unrolling of the Baugh-Wooley multiplier was more than the other two multipliers, although the sign-magnitude multiplier still consumes less power even with unrolling. In the comparison between the two's complement multipliers, the Baugh-Wooley multiplier performed better than the Booth-Encoding multiplier over all correlations when unrolling was used.

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	13.2	13.5	13.7	13.6	13.5	13.6	13.4	13.4	13.6	13.4
2	9.29	9.53	9.74	9.63	9.41	9.51	9.42	9.57	9.59	9.57
3	6.26	6.36	6.45	6.42	6.35	6.35	6.33	6.31	6.35	6.32
4	4.71	4.77	4.96	4.83	4.74	4.72	4.75	4.79	4.84	4.74

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	13.5	13.5	13.2	13.3	13.3	13.4	13.4	13.3	13.0
2	9.56	9.58	9.43	9.37	9.48	9.38	9.47	9.40	9.29
3	6.36	6.35	6.32	6.25	6.31	6.33	6.33	6.33	6.20
4	4.82	4.76	4.75	4.72	4.73	4.68	4.74	4.73	4.66

Table 4.1: Power simulation results in mWatts for vertical unrolling with $\sigma = 1000$ sign-magnitude data using a sign-magnitude array multiplier.

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	18.7	18.8	18.8	18.7	18.7	18.7	18.6	18.6	18.7	18.6
2	12.6	12.8	12.9	13.1	13.3	13.3	13.4	13.3	13.5	13.5
3	9.15	9.18	9.13	9.01	8.98	8.89	8.86	8.87	8.94	8.90
4	6.43	6.52	6.57	6.66	6.67	6.70	6.69	6.68	6.73	6.70

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	18.6	18.6	18.6	18.6	18.5	18.5	18.4	18.4	18.4
2	13.3	13.5	13.4	13.4	13.2	13.2	13.0	12.9	12.8
3	8.92	8.91	8.86	8.86	8.82	8.88	8.71	8.75	8.57
4	6.70	6.76	6.70	6.71	6.64	6.65	6.59	6.57	6.46

Table 4.2: Power simulation results in mWatts for vertical unrolling with $\sigma = 1000$ two's complement data using a Booth-Encoding multiplier

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	21.2	21.5	21.3	20.5	20.6	20.1	19.7	19.6	20.0	19.6
2	11.2	11.6	11.9	12.2	12.6	12.5	12.8	12.5	13.0	13.0
3	9.01	9.11	9.02	8.67	8.65	8.45	8.45	8.36	8.57	8.48
4	5.82	6.02	6.12	6.31	6.34	6.34	6.47	6.35	6.49	6.42

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	19.6	19.2	19.6	19.5	18.7	18.8	17.9	18.0	17.9
2	12.8	12.8	12.8	12.8	12.4	12.5	12.0	11.8	11.7
3	8.52	8.44	8.46	8.49	8.30	8.46	8.04	8.12	7.92
4	6.41	6.45	6.44	6.49	6.24	6.35	6.16	6.11	5.99

Table 4.3: Power simulation results in mWatts for vertical unrolling for $\sigma = 1000$ two's complement data using a Baugh-Wooley array multiplier

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	8.91	9.04	8.88	9.09	9.14	8.03	9.19	9.13	9.23	8.81
2	6.44	6.47	6.23	6.48	6.50	6.49	6.49	6.44	6.44	6.38
3	4.30	4.39	4.23	4.37	4.36	4.30	4.35	4.32	4.40	4.19
4	3.23	3.21	3.12	3.23	3.27	3.23	3.25	3.22	3.24	3.20

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	9.15	9.13	9.19	9.05	9.06	9.09	8.99	8.91	8.90
2	6.57	6.54	6.62	6.50	6.37	6.53	6.51	6.38	6.34
3	4.34	4.36	4.37	4.29	4.31	4.34	4.33	4.32	4.29
4	3.30	3.27	3.31	3.21	3.17	3.25	3.25	3.20	3.20

Table 4.4: Power simulation results in mWatts for vertical unrolling with $\sigma = 100$ two's complement data using a sign-magnitude multiplier

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	16.9	16.9	16.7	16.8	16.8	16.7	16.7	16.7	16.6	16.6
2	10.9	11.3	11.4	11.5	11.6	11.8	11.9	11.9	11.7	11.9
3	8.50	8.24	8.12	8.10	7.97	7.94	7.89	7.93	7.93	7.88
4	5.60	5.78	5.81	5.85	5.96	5.89	5.97	5.92	5.87	5.98

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	16.6	16.7	16.6	16.5	16.5	16.6	16.4	16.3	16.3
2	12.0	12.0	11.9	11.7	11.7	11.5	11.6	11.4	10.9
3	7.92	8.00	7.92	7.85	7.83	7.79	7.78	7.65	7.40
4	5.96	6.00	5.93	5.89	5.91	5.85	5.91	5.78	5.60

Table 4.5: Power simulation results in mWatts for vertical unrolling with $\sigma = 100$ two's complement data using a Booth-Encoding multiplier

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	19.4	18.5	17.7	17.8	17.5	17.1	16.9	16.8	16.3	16.5
2	8.80	9.21	9.40	9.40	9.78	10.1	10.3	10.3	10.0	10.5
3	7.95	7.46	7.19	7.15	6.94	6.89	6.82	6.91	6.84	6.91
4	4.61	4.90	4.89	4.90	5.14	5.06	5.21	5.15	4.99	5.27

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	16.3	16.0	16.1	15.7	15.2	15.2	14.7	14.2	13.7
2	10.5	10.5	10.2	10.1	10.0	9.90	9.73	9.35	8.69
3	6.83	6.96	6.84	6.73	6.67	6.67	6.59	6.37	5.97
4	5.18	5.27	5.11	5.08	5.10	5.10	5.04	4.87	4.56

Table 4.6: Power simulation results in mWatts for vertical unrolling with $\sigma = 100$ two's complement data using a Baugh-Wooley multiplier

$\sigma = 1000$	No. of Unrollings		
Multiplier	2	3	4
Sign-Magnitude Array	28.5-29.0%	52.1-53.3%	63.8-65.3%
Two's Complement Booth-Encoding	27.4-32.6%	51.1-53.4%	63.7-65.6%
Two's Complement Baugh-Wooley	33.0-47.2%	54.9-58.0%	66.1-72.5%
$\sigma = 100$	No. of Unrollings		
Multiplier	2	3	4
Sign-Magnitude Array	19.2-30.2%	46.5-52.7%	59.8-65.0%
Two's Complement Booth-Encoding	27.7-35.5%	45.4-54.6%	64.0-66.9%
Two's Complement Baugh-Wooley	33.8-50.2%	55.1-62.8%	65.7-73.5%

Table 4.7: Range of power reduction for various multipliers, data sets, and vertical unrollings relative to conventional convolution multiplication with no unrollings.

4.2 Diagonal Unrolling

4.2.1 Concept

Another method that was used to reduce transition activity also used block processing in the same manner as vertical unrolling. In this method, the data was unrolled to compute several outputs simultaneously, as before, but now the signal instead of the coefficient was kept constant for as many multiplications as possible.[4] Thus, for an unrolling of two, the order of multiplications, once again denoted by the bold superscript numbers, is

$$y_0 = c_0\overset{\mathbf{2}}{x}_0 + c_1\overset{\mathbf{4}}{x}_{-1} + c_2\overset{\mathbf{6}}{x}_{-2} \tag{4.6}$$

$$y_1 = c_0\overset{\mathbf{1}}{x}_1 + c_1\overset{\mathbf{3}}{x}_0 + c_2\overset{\mathbf{5}}{x}_{-1}\,. \tag{4.7}$$

One observation that can be made is that for diagonal unrolling of order two, as shown here, the signal is not only held constant for two multiplications, but the coefficients are held constant for two multiplications as well. Furthermore, when the signal does change, it is only by a single sample. This is taken advantage of when data is in a form where a high first-order correlation decreases transition activity. These added benefits only occur for diagonal unrolling of order two. In general, however, diagonal unrolling is useful when there is greater transition activity in the data than in the set of coefficients, since the data input is held constant for successive multiplications.

Again, as in all cases of block processing, the cost of holding the inputs to the multiplier constant is the requirement of additional storage elements.

4.2.2 Results

The tables in this subsection show the effects of diagonal unrolling when performed on various data sets and multipliers. The simulations were run in the same manner as in the case of vertical unrolling. The power reduction through diagonal unrollings of 2, 3, and 4 can be found in Table 4.14.

The nature of the results as based on the type of multiplier are as seen before. There is little dependence of power on the correlation of the data in both the sign-magnitude and Booth-Encoding multiplier. However, once again, power is found to be inversely correlated with the correlation of the data in the Baugh-Wooley multiplier.

In contrast with vertical unrolling, increasing the number of diagonal unrolls does not cause power to go down monotonically. In all cases, there is a significant decrease in power when diagonal unrolling of order 2 is used. Power consumption then goes *up* when additional unrollings are used. As mentioned in Section 4.2.1, in diagonal unrolling of order 2, the data input is kept constant between one multiply and the next; in the subsequent multiplication, the FIR coefficient is held constant. Thus, only in diagonal unrolling of order 2 will at least one input to the multiplier be kept constant between successive multiplications. This structure is no longer taken advantage of when additional unrollings are performed. This characteristic is most pronounced with a Booth-encoding multiplier. This can be attributed to the fact that even in the case of high correlation, there will still be significant transitions in this multiplier. As seen in Appendix A, in the Booth-encoding multiplier, several adjacent bits of the multiplier must be identical for the state of the multiplier to remain the same.

Once again, the performance of the sign-magnitude multiplier is significantly better than either of the two's complement multipliers. In terms of power reduction, summarized in Table 4.14, the sign-magnitude multiplier is only outperformed by the Baugh-Wooley multiplier, and even then not universally. The Baugh-Wooley multiplier, interestingly, reduces power more as correlation increases in diagonal unrolling of order 2, yet increases power consumption in diagonal unrollings of order 3 and 4. This is a result of the decreasing dependence of power on correlation as unrollings increase in the Baugh-Wooley multiplier. Regardless, the Baugh-Wooley outperforms the Booth-Encoding multiplier in all cases.

Comparing the two types of unrolling, it is evident that vertical unrolling reduces power consumption more than diagonal unrolling. In diagonal unrolling, the signal input jumps two samples between diagonals. In vertical unrolling, the signal input always changes by only a single sample even between vertical columns. This results in less transitions and less power.

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	13.2	13.5	13.7	13.6	13.5	13.6	13.4	13.4	13.6	13.4
2	9.55	9.73	9.87	9.87	9.81	9.83	9.68	9.73	9.91	9.75
3	10.3	10.6	10.6	10.6	10.5	10.6	10.5	10.4	10.6	10.5
4	10.2	10.4	10.5	10.5	10.4	10.4	10.3	10.3	10.4	10.3

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	13.5	13.5	13.2	13.4	13.3	13.4	13.4	13.3	13.0
2	9.85	9.82	9.62	9.69	9.63	9.72	9.70	9.62	9.45
3	10.5	10.5	10.4	10.4	10.4	10.4	10.4	10.4	10.2
4	10.4	10.4	10.2	10.2	10.2	10.2	10.3	10.2	10.2

Table 4.8: Power simulation results in mWatts for diagonal unrolling with $\sigma = 1000$ data using a sign-magnitude array multiplier

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	18.7	18.8	18.9	18.8	18.8	18.7	18.6	18.6	18.7	18.6
2	14.5	14.4	14.4	14.2	14.1	14.1	13.8	13.8	13.8	13.7
3	17.0	17.2	17.2	17.1	17.1	17.1	17.0	16.9	17.1	17.0
4	17.1	17.3	17.3	17.2	17.2	17.2	17.1	17.1	17.2	17.1

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	18.6	18.6	18.6	18.5	18.4	18.4	18.5	18.3	18.3
2	13.6	13.6	13.5	13.4	13.2	13.2	13.2	12.8	12.6
3	17.0	17.0	17.0	16.9	16.9	16.9	16.9	16.8	16.9
4	17.1	17.1	17.1	17.1	17.1	17.0	17.1	16.9	17.0

Table 4.9: Power simulation results in mWatts in diagonal unrolling with $\sigma = 1000$ using a Booth-Encoding multiplier

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	21.7	21.8	21.6	20.9	20.9	20.5	20.0	19.7	20.2	19.6
2	14.8	14.7	14.6	14.2	14.1	13.8	13.4	13.3	13.5	13.2
3	14.8	15.0	15.0	14.6	14.7	14.5	14.3	14.1	14.6	14.2
4	14.5	14.8	14.8	14.4	14.5	14.3	14.1	13.9	14.4	14.0

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	19.5	19.1	19.4	19.2	18.1	18.4	17.5	17.4	17.4
2	13.0	12.7	12.9	12.7	12.0	12.2	11.6	11.5	11.3
3	14.2	14.0	14.2	14.1	13.6	13.8	13.3	13.3	13.6
4	14.0	13.9	14.1	14.1	13.5	13.8	13.3	13.3	13.8

Table 4.10: Power simulation results in mWatts in diagonal unrolling with $\sigma = 1000$ using a Baugh-Wooley multiplier

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	8.91	9.04	8.88	9.09	9.14	8.93	9.19	9.13	9.23	8.81
2	6.47	6.59	6.50	6.66	6.67	6.56	6.74	6.71	6.79	6.50
3	7.02	7.11	6.89	7.10	7.13	6.95	7.11	7.04	7.18	6.82
4	6.92	7.03	6.80	7.02	7.04	6.83	7.00	6.93	7.09	6.72

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	9.15	9.13	9.19	9.05	9.06	9.09	8.99	8.91	8.90
2	6.74	6.70	6.73	6.63	6.65	6.67	6.58	6.52	6.45
3	7.07	7.08	7.16	7.03	7.04	7.06	7.02	6.96	7.02
4	6.94	6.96	7.05	6.92	6.94	6.96	6.91	6.87	6.93

Table 4.11: Power simulation results in mWatts for diagonal unrolling with $\sigma = 100$ data using a sign-magnitude array multiplier

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	17.0	17.0	16.8	16.8	16.9	16.8	16.8	16.8	16.6	16.6
2	13.6	13.4	13.0	13.0	12.8	12.7	12.6	12.4	12.2	12.1
3	15.5	15.5	15.4	15.4	15.4	15.3	15.3	15.4	15.2	15.3
4	15.6	15.6	15.5	15.6	15.6	15.5	15.5	15.5	15.4	15.5

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	16.6	16.6	16.6	16.5	16.4	16.4	16.3	16.2	16.2
2	12.1	11.9	11.9	11.8	11.5	11.3	11.2	10.9	10.6
3	15.3	15.3	15.3	15.2	15.1	15.2	15.1	15.1	15.1
4	15.4	15.4	15.5	15.3	15.3	15.4	15.3	15.3	15.3

Table 4.12: Power simulation results in mWatts for diagonal unrolling with $\sigma = 100$ data using a Booth-encoding multiplier

	Correlation									
No. of Unrolls	-.9	-.8	-.7	-.6	-.5	-.4	-.3	-.2	-.1	0
1	19.8	18.1	18.2	18.4	18.1	17.7	17.5	17.1	16.3	16.7
2	13.2	12.7	12.0	12.1	11.9	11.6	11.4	11.0	10.6	10.7
3	12.6	12.3	11.8	12.0	12.0	11.7	11.7	11.7	11.2	11.5
4	12.3	12.0	11.6	11.8	11.8	11.5	11.6	11.5	11.0	11.5

	Correlation								
No. of Unrolls	.1	.2	.3	.4	.5	.6	.7	.8	.9
1	16.1	15.7	15.6	15.4	14.6	14.5	14.0	13.5	13.1
2	10.5	10.1	10.0	9.93	9.34	9.16	8.84	8.46	8.06
3	11.2	11.2	11.1	10.9	10.6	10.8	10.5	10.3	10.3
4	11.1	11.1	11.0	10.9	10.6	10.8	10.6	10.5	10.4

Table 4.13: Power simulation results in mWatts for diagonal unrolling with $\sigma = 100$ data using a Baugh-Wooley multiplier

$\sigma = 1000$	No. of Unrollings		
Multiplier	2	3	4
Sign-Magnitude Array	27.0-28.0%	21.5-22.7%	21.5-23.4%
Two's Complement Booth-Encoding	22.5-26.3%	8.5-9.1%	8.0-8.6%
Two's Complement Baugh-Wooley	31.8-35.1%	21.8-31.8%	20.7-33.2%
$\sigma = 100$	No. of Unrollings		
Multiplier	2	3	4
Sign-Magnitude Array	26.2-27.5%	21.1-22.9%	22.1-24.2%
Two's Complement Booth-Encoding	20.0-34.6%	6.8-8.9%	5.6-8.2%
Two's Complement Baugh-Wooley	29.8-38.5%	21.4-36.4%	20.6-37.9%

Table 4.14: Range of power reduction for various multipliers, data sets, and diagonal unrollings relative to conventional convolution multiplication with no unrollings.

4.3 Unrolling Applied To Toeplitz Matrices

4.3.1 Concept

A *toeplitz matrix* is a symmetric matrix whose diagonals are equal. A well-used example of a toeplitz matrix in speech processing is the autocorrelation matrix, **R**, used in linear predictive coding (LPC):

$$\mathbf{R}\vec{\alpha} = \begin{pmatrix} r(0) & r(1) & r(2) & \cdots & r(p-1) \\ r(1) & r(0) & r(1) & \cdots & r(p-2) \\ r(2) & r(1) & r(0) & \cdots & r(p-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r(p-1) & r(p-2) & r(p-3) & \cdots & r(0) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_p \end{pmatrix} = \begin{pmatrix} r(1) \\ r(2) \\ r(3) \\ \vdots \\ r(p) \end{pmatrix}, \tag{4.8}$$

where

$$r(k) = \sum_{m=0}^{N-1} s(m)s(m+k), \tag{4.9}$$

and $s(n)$ is a discrete-time speech signal.

Briefly, LPC analysis looks at a section of speech data and attempts to estimate the data with

$$\tilde{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k), \tag{4.10}$$

where s(n) is the actual speech data. Equation 4.8 can be solved to find $\vec{\alpha}$. [9]

4.3.2 Simulation and Results

Since array multiplication is simply a series of multiplications, the structure inherent in a Toeplitz matrix allows for judicious ordering of the multiplications to help reduce power consumption within the multiplier. For the simulation, an actual speech waveform, $s(n)$, was used to generate the autocorrelation matrix **R** and determine the LPC coefficient vector, $\vec{\alpha}$. Then, in a simulated experiment, the series of multiplications inherent in the matrix multiplication $\mathbf{R}\vec{\alpha}$ was performed, first without taking advantage of the Toeplitz structure and then with. The symmetries were utilized such that all the multiplications involving a diagonal of the matrix **R** were performed at once. The multiplications were ordered starting from the lower left corner of the Toeplitz matrix – a diagonal with only a single term. The

multiplications then progressed to the center diagonal and then to the upper right corner of the matrix, with the number of multiplications along each diagonal first growing and then shrinking back to a single term. Thus, depending on the diagonal, a different number of inputs were kept constant to the multiplier.

The experiment was performed with a sign-magnitude multiplier first with the correlations as the x input to the multiplier and the LPC coefficients as the y input, and then with these inputs switched. In these examples $\mathbf{R}$ was a 16×16 matrix. The results can be seen in Table 4.15.

These results show that there is significant room for reducing the power consumption by a simple reordering of the multiplications in matrix multiplications involving Toeplitz matrices. A reduction of 17-18% was realized by considering the Toeplitz symmetry in performing the multiplications. The reduction in power would be greater as larger Toeplitz matrices are considered.

Multiplier	Method	Power (mwatts)
Sign-Magnitude	No regard to Toeplitz symmetry (xy)	4.10
Sign-Magnitude	Using Toeplitz symmetry (xy)	3.33 (18.8%)
Sign-Magnitude	No regard to Toeplitz symmetry (yx)	6.62
Sign-Magnitude	Using Toeplitz symmetry (yx)	5.79 (17.1%)

Table 4.15: Power consumption in simulation of Toeplitz matrix multiplications

Chapter 5

Coefficient Perturbation

5.1 Introduction

The methods that have been discussed in Chapter 4 rely on block processing. Thus, the power reductions all come at the cost of additional memory. There may be times when excess memory is not available or prohibitively expensive. In these cases, other methods must be utilized in order to reduce power. Although generally one does not have any control over the input signal in an FIR multiplication, often, in non-adaptive filtering, the actual FIR filter is known beforehand. Accordingly, if the transition activity of the filter can be reduced in some manner, the power consumption within the multiplier can also be reduced, since it has been established that power consumption is proportional to the number of transitions in the inputs. Perturbation of the coefficients is one method that can reduce the transition activity. By selectively changing bits of the filter coefficients, the number of transitions between the successive coefficients can be reduced. It will be seen that small perturbations can be made to the coefficients without significant changes to the frequency response of the filter.

As long as obtaining an exact result is not an important concern, the coefficients of a FIR filter can be perturbed without sacrificing the frequency response of the filter. It has been found to be more useful to look at the actual bit representation of the coefficients rather than to use a systems-driven signal-processing approach. This observation is motivated by the randomness of the low-order bits of even when the data is correlated (Section 3.2). Since the perturbation is intended to preserve the general frequency response of the filter, it is these low-order bits that will be perturbed. Thus, in order to be effective, any proposed

routine must address the bit representations themselves.

5.2 Previous Research

The idea of perturbing coefficients in order to reduce transition activity and thus power consumption was addressed by Mehendale, Sherlekar, and Venkatesh in *VLSI Signal Processing, VIII* [6]. The algorithm they created was designed to reduce the number of transitions in the coefficients while maintaining the response characteristics that were desired.

The optimization algorithm has two stages. The first stage uniformly scales the coefficients, $\{c_i\}$, constraining the gain to $\pm$3db. This scaling does not distort the overall frequency response of the filter, but only introduces a gain term. The second stage of the algorithm perturbs individual coefficients in an iterative process. In each step of the process, a new set of coefficients, $\{c_i'\}$ is potentially generated with only a single coefficient perturbed. In order to create this new set, $2N$ sets of coefficients are first created in the following manner. First, a coefficient is selected. Then this coefficient is incremented until the number of transitions between it and its adjacent coefficients decreases. This produces a candidate set of coefficients. The filter response of these coefficients is then tested to see if it fits the desired constraints of passband ripple and stopband attenuation. If the filter passes these tests, then the coefficients are rated by the measure,

$$\gamma = \frac{Pdb_{req} - Pdb}{Pdb_{req}} + \frac{Sdb - Sdb_{req}}{Sdb_{req}} \times HD_{red}, \tag{5.1}$$

where HD_{red} is the reduction in the total Hamming Distance (or number of transitions) for the new set of coefficients from the current best set of coefficients, Pdb_{req} is the required passband ripple, Sdb_{req} is the required stopband attenuation, Pdb is the passband ripple of the new set of coefficients, and Sdb is the stopband attenuation of the new set of coefficients. If the set did not fulfill the filter constraints, γ is set to zero. For the same coefficient, another potential set of coefficients is constructed by decrementing the selected coefficient to produce a set with fewer transitions. These procedures are repeated for each of the N coefficients to produce the $2N$ sets of coefficients. From these candidate set of coefficients, the new set of coefficients is selected by choosing the set that had the largest non-zero measure, γ. This new set becomes the starting point for the next step of the iteration. The iteration continues until no non-zero γs were available.

The algorithm allows for the option to maintain any existing linear phase. With this requirement, coefficients are perturbed in pairs (c_i and c_{N-1-i}) and only the first $\frac{N+1}{2}$ coefficients are analyzed in each stage of the iteration.

The results of applying this algorithm to several Parks-McClellan filters with and without the linear phase constraint produced up to a 36% reduction in transition activity.

The algorithm that will be described in this chapter has some similarities to Mehendale, et. al. There will be a scaling component before any perturbation and the perturbation will exhibit an iterative nature as well. The performance of the algorithm will also be tested against Parks-McClellan filters. However, the algorithm will be different in several respects. The effect of the rigidity of the filter constraints on the algorithm will be addressed more thoroughly. The algorithm itself approaches the problem by looking at larger subsets of the data. Each step of the iteration also uses the original set of coefficients as a starting point. No arbitrary measure will be used to determine the goodness of a set of coefficients; the number of transitions will solely determine a new set of coefficients. Finally, the validity of the effect that reducing transitions through perturbation has on power will be explored and tested.

5.3 The Problem

The coefficient perturbation problem can be stated as follows:
Given a N-tap FIR filter with coefficients $\{c_i\}$ that satisfy the filter response in terms of an initial passband ripple and initial stopband attenuation, find a new set of coefficients $\{c_i'\}$ such that the number of transitions between successive coefficients is reduced while satisfying the desired filter characteristics in terms of a final passband ripple and a final stopband attenuation. In addition, if desired, any linear phase characteristics will be maintained.

5.4 Routine Overview

Figure 5-1 shows a flow diagram of the proposed routine. There are three major sections to the routine. The first is the filter creation stage where an initial set of coefficients based on the desired parameters is created. The remaining stages seek to reduce the transitions in this initial set of coefficients. The second stage is an optional scaling stage which scales the data within an allowable range. The third and final stage perturbs the coefficients to

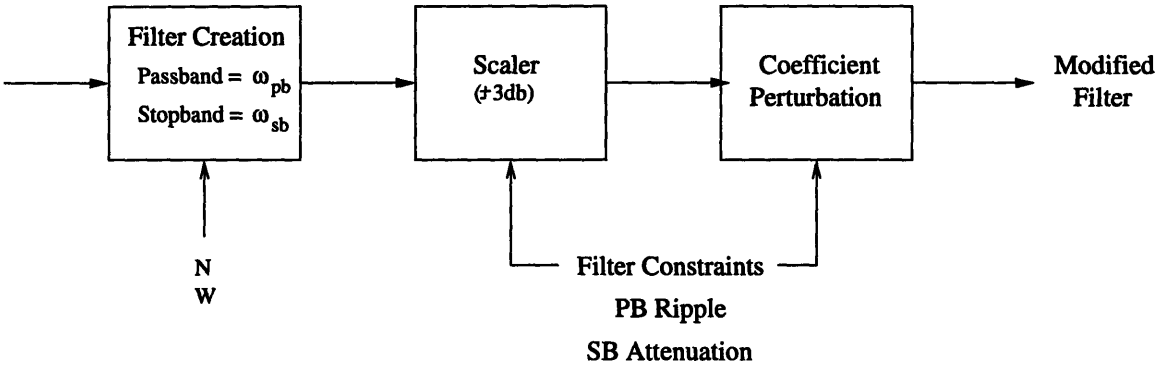

Figure 5-1: Flow diagram for coefficient perturbation routine

reduce transition activity under the passband and stopband constraints.

5.4.1 Filter Creation

The filters were created using a Park-McClellan algorithm to create an optimum FIR filter with a passband frequency of ω_p and a stopband frequency of ω_s. The filter is optimal in that it has the smallest maximum weighted approximation error with the given passband and stopband. Two additional parameters were used to control the passband ripple and stopband attenuation of the filter. These were N, the number of coefficients, or taps, of the filter, and W, the weight given to the stopband error relative to passband error. [7]

5.4.2 Filter Constraints

An important factor in modifying the coefficients is the flexibility of the passband ripple and stopband attenuation constraints. The initial filter was created such that initial values of the passband ripple and stopband attenuation were within the desired constraints. However, due to the optimality of the initial filter, any perturbation of the coefficients will inevitably have a larger passband ripple and/or a smaller stopband attenuation. Therefore, in order for the routine to be effective, the design constraints have "gray zones", where a filter will have a larger passband ripple and smaller stopband attenuation than the initial filter created by the Park-McClellan algorithm. As the size of these "gray zones" increase, there is greater flexibility in reducing bit transitions, as will be seen in Section 5.6.

5.4.3 Scaling

The first step in reducing the transition activity of the coefficient involves uniformly scaling the data by some constant K. The range of this constant is restricted to ±3dB. Scaling of the coefficients, $\{c_i\}$, preserves all of the original frequency characteristics of the filter, only introducing a gain factor. This is seen by applying the constant scaling K to each coefficient in Equation 1.1:

$$y'_k = \sum_{i=0}^{N-1} Kc_i x_{k-i} = K \sum_{i=0}^{N-1} c_i x_{k-i} = Ky_k. \tag{5.2}$$

The constant, K, is not restricted to integers, so the bit pattern of the binary representation of the scaled set of coefficients, $\{Kc_i\}$, is different and can have a reduced number of transitions. The optimal scaling constant, K, is obtained by selecting the constant within the range ±3db that minimizes the number of coefficient transitions. This modified set of coefficients, $\{Kc_i\}$, is used as the starting point of the coefficient perturbation stage. In some cases a gain may not be tolerated and the perturbation problem is approached by omitting the scaling component (or by constraining K to be 1). Furthermore, testing of any proposed filters was done after accounting for the scaling factor. Scaling has the benefit of reducing the number of transitions without altering the frequency response. Scaling then allows the perturbation module to use the constraints to further reduce transitions. It must also be noted that scaling is a reversible process. If desired, the final output of the perturbed filter may be multiplied by K^{-1} to undo the effect of the scaling.

5.4.4 Perturbation

In coefficient perturbation, the lower k bits of the coefficients are extracted, with k starting from 1 and ranging to all of the non-sign bits. First, the lower k bits of the complete set of coefficients, $\{c_i\}$, is analyzed. These low k are converted back to a decimal equivalent and averaged. Then all the low k bits of the coefficients are replaced by the binary equivalent of the average. If the transitions of the new set of coefficients is found to be less than the previous best set, and the set fulfills the frequency requirements, the new set becomes the best set. If not, the procedure is successively repeated with subsets of the coefficient set. When a best set that works for a given value of k is found or the transition of a set of potential coefficients is less than the current best set, the routine is iterated with the low

$k+1$ bits. This is because any further sets of coefficients with the same value of k cannot do better than the current best set. The routine is also iterated when a potential set with subsets of only two coefficients fails, since all perturbation opportunities are exhausted for that given value of k.

5.5 Pseudo Code

In this section, pseudo code describing the coefficient perturbation routine is presented.

$TA_{best} = TA(\{c_i\})$

$\{c_{i,best}\} = \{c_i\}$

for the low k order bits ($k = 1, \text{bit width}$) {

    foo=TRUE

    for $l = N^*$ to 2 AND while foo is TRUE {

    Divide coefficients into subsets of length l.

        for each subset {

            Average the last k bits of each original coefficient in subset.

            Generate a new subset of coefficients by replacing last k original

                bits of each coefficient in subset with the average.*

        }

        The potential set of coefficients $\{c_i'\}$ consists of all of the modified subsets

            and any unmodified coefficients.

        Compute the passband ripple ($PBrip'$) and stopband attenuation ($SBatten'$).

        if(($PBrip' < PBrip_{req}$) AND ($SBatten' < SBatten_{req}$) AND ($TA(\{c_i'\}) < TA_{best}$)) {

            $TA_{best} = TA(\{c_i'\})$

            $\{c_{i,best}\} = \{c_i'\}$

            foo=FALSE

        }

    }

* Under Linear Phase constraint, restrict routine to first $\frac{N}{2}$ coefficients and apply any perturbations to coefficient c_i to coefficient c_{N-1-i} as well.

It must be noted that when the number of coefficients, N, is not a multiple of l, the length of the subsets, there will be some coefficients that will not fit in a subset. These coefficients

are kept unaltered in the modified set of coefficients. Without the linear phase constraint, the unmodified coefficients will be at the end of the set of coefficients. However, with the linear phase constraint, the unmodified coefficients will be symmetrically distributed in the middle of the set of coefficients. Furthermore, even when phase is not a concern, the linear phase constraint is applied first to see if a solution that preserves any symmetries is possible.. Then, the routine is applied without the constraint. This method allows one to ascertain whether a better solution that maintains linear phase is obtainable.

	Initial Filter	Step 1	Step 2	Step 3a	Step 3b	
c_0	001111	001111	001110	001101	001100	
c_1	010100	010101	010110	010101	010100	
c_2	111001	111001	111010	111101	111100	
c_3	011110	011111	011110	011101	011110	...
c_4	011110	011111	011110	011101	011110	
c_5	111001	111001	111010	111101	111100	
c_6	010100	010101	010110	010101	010100	
c_7	001111	001111	001110	001101	001100	
Number of Transitions:	24	18	14	~~10~~	12	

Figure 5-2: An example of the coefficient perturbation routine.

An artificial example working through the routine is shown in Figure 5-2. The Initial Filter is generally created using the Park-McClellan algorithm. In this example, an arbitrary set of coefficients is chosen for illustration purposes. This initial filter has 24 transitions. Since we desire to maintain linear phase, the coefficients will be perturbed in pairs. Thus, the first $\frac{N}{2} = 4$ coefficients (i.e. $c_0 - c_4$) are analyzed for averaging purposes. Step 1 replaces the lowest bit with the average of the bits (rounded). This filter is found to fit desired filter constraints and reduces the number of transitions to 18. Step 2 looks at the two lowest order bits and replaces them with their average. This filter works and has only 14 transitions. Step 3a now looks at the three lowest bits. However, this filter is found to violate the desired constraints, and we disregard this filter even though there are fewer transitions. Step 3b continues to look at the lowest three bits, but now looks at subsets of length 3. This filter is found to work, and this solution becomes the new set of coefficients with 12 transitions. Note that the unperturbed coefficients are the fourth and fifth coefficients, thereby maintaining symmetry and linear phase. The routine would continue to look at a

larger number of the low-order bits. In the case where none of these filters work and provide a better set of coefficients, the solution of Step 3b would become the final modified filter with linear phase maintained.

5.6 Results

Tables 5.1 - 5.3 describe 5 lowpass filters and the results of applying the coefficient perturbation routine to the filters. The routine is performed with and without including the scaling module. Additionally, a constraint of maintaining linear phase is applied. In many cases, the results only show a case with linear phase. These instances did not have a perturbation with fewer transitions without the linear phase constraint. For each filter, several groups of runs were made with each group having different routine constraints. The percentages attached to the resultant number of transitions represents the reduction in transition activity of the filter. Figure 5.6 shows the frequency response of the original Lowpass Filter 1 and the perturbed filter, namely the first perturbation that yielded a 56% reduction in transitions.

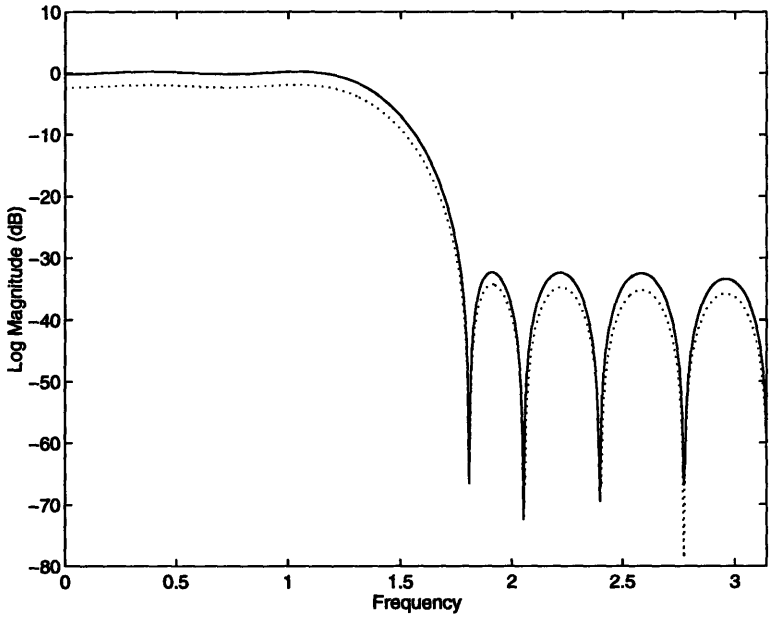

Figure 5-3: Frequency response of original Lowpass Filter 1 (solid line) and perturbed filter (dotted line). The offset gain between the two filters is due to the scaling. A post-convolution multiplication can remove the offset.

Lowpass Filter 1

Sampling Frequency=16kHz — Number of Coefficients = 16
Passband Frequency=3kHz — Passband Ripple = .2092
Stopband Frequency=4.5kHz — Stopband Attenuation = 30.84
Initial Number of Transitions = 100

Scaling?	Lin. Phase?	Routine Constraints		Number of Transitions
		PB Ripple	SB Attenuation	
No	Yes	.210	30	80 (20.0%)
Yes	Yes	.210	30	44 (56.0%)
No	Yes	.215	30	66 (34.0%)
Yes	Yes	.215	30	44 (56.0%)

Lowpass Filter 2

Sampling Frequency=16kHz — Number of Coefficients = 50
Passband Frequency=3kHz — Passband Ripple = .0636
Stopband Frequency=4kHz — Stopband Attenuation = 62.49
Initial Number of Transitions = 224

Scaling?	Lin. Phase?	Routine Constraints		Number of Transitions
		PB Ripple	SB Attenuation	
No	Yes	.075	62	190 (15.2%)
Yes	Yes	.075	62	196 (12.5%)
No	Yes	.1	60	190 (15.2%)
Yes	Yes	.1	60	170 (24.1%)
Yes	No	.1	60	164 (26.8%)

Table 5.1: Results of routine applied to various lowpass filters

From the results, relaxing the routine constraints allows for a greater reduction in transition activity. Relaxed constraints imply a larger "gray zone". More more possible filters exist with relaxed constraints, so there is a greater opportunity to find a solution that reduces transitions more. In most cases, utilization of the scaling module improves the effectiveness of the routine.

Generally, there is less improvement when the routine is applied to a filter with a odd number of coefficients then when it is to an even-length filter. Furthermore, with odd-length filters, the linear phase constraint became more significant.

5.7 Power Analysis

The effectiveness that the reduction of transition activity by the coefficient perturbation had on power was tested using Lowpass Filter 1. A simple convolution with a signal was

performed with the filter, before and after the coefficients were perturbed. The best perturbation of Filter 1 was chosen, namely the one with a passband ripple constraint of .210 and a stopband attenuation of 30dB. The signals were the $\rho = 0$ streams taken from the $\sigma = 1000$ and $\sigma = 100$ data sets. The results of the power simulations are shown in Table 5.7. The power simulations were done using both PYTHIA and POWERMILL. POWERMILL is a more accurate transistor-level power simulator. The results indicate that the coefficient perturbations do have an impact on power, reducing power consumption by about 25% in the PYTHIA simulation. The results were even more dramatic using POWERMILL, reducing power by over 30%. The power values in POWERMILL are lower than those in PYTHIA since PYTHIA is more conservative in its power estimation routines.

Lowpass Filter 3

Sampling Frequency=10kHz
Passband Frequency=1.8kHz
Stopband Frequency=2.5kHz
Initial Number of Transitions = 182

Number of Coefficients = 41
Passband Ripple = .0956
Stopband Attenuation = 60.08

		Routine Constraints		Number of
Scaling?	Lin. Phase?	PB Ripple	SB Attenuation	Transitions
No	Yes	.100	60	156 (14.3%)
Yes	No	.100	60	148 (18.7%)
Yes	Yes	.100	60	158 (13.2%)
No	Yes	.150	60	150 (17.6%)
Yes	No	.150 (.1003)	60	132 (27.5%)
Yes	Yes	.150	60	158 (13.2%)
No	Yes	.100	58	156 (14.3%)
Yes	No	.100	58	148 (18.7%)
Yes	Yes	.100	58	158 (13.2%)
No	Yes	.150	58	142 (22.0%)
Yes	Yes	.150	58	158 (13.2%)
Yes	No	.150	58	132 (27.5%)

Lowpass Filter 4

Sampling Frequency=12kHz
Passband Frequency=2kHz
Stopband Frequency=3kHz
Initial Number of Transitions = 124

Number of Coefficients = 28
Passband Ripple = .1169
Stopband Attenuation = 46.01

		Routine Constraints		Number of
Scaling?	Lin. Phase?	PB Ripple	SB Attenuation	Transitions
No	Yes	.12	45	96 (22.6%)
Yes	Yes	.12	45	84 (32.3%)
No	Yes	.13	45	88 (29.0%)
Yes	Yes	.13	45	84 (32.3%)
No	Yes	.13	44	82 (33.9%)
Yes	Yes	.13	44	84 (32.3%)

Table 5.2: Results of routine applied to various lowpass filters

Lowpass Filter 5

Sampling Frequency=10kHz
Passband Frequency=2kHz
Stopband Frequency=3kHz
Initial Number of Transitions = 150

Number of Coefficients = 29
Passband Ripple = .0458
Stopband Attenuation = 57.90

		Routine Constraints		Number of
Scaling?	Lin. Phase?	PB Ripple	SB Attenuation	Transitions
No	Yes	.05	57	130 (13.3%)
No	No	.05	57	132 (12.0%)
Yes	Yes	.05	57	112 (25.3%)
Yes	No	.05	57	100 (33.3%)
No	Yes	.05	56	126 (16.0%)
No	No	.05	56	124 (17.3%)
Yes	Yes	.05	56	112 (25.3%)
Yes	No	.05	56	96 (36.0%)
No	Yes	.055	57	102 (32.0%)
No	No	.055	57	100 (33.3%)
Yes	Yes	.055	57	112 (25.3%)
Yes	No	.055	57	98 (34.7%)

Table 5.3: Results of routine applied to various lowpass filters

	Power (mWatts)(PYTHIA)		Power (mWatts)(POWERMILL)	
Filter	$\sigma = 100$	$\sigma = 1000$	$\sigma = 100$	$\sigma = 1000$
Original Filter	5.43 (100%)	8.50 (100%)	3.63 (100%)	6.32 (100%)
Filter w/perturbed coefficients	4.20 (77.3%)	6.40 (75.3%)	2.38 (65.6%)	4.27 (67.6%)

Table 5.4: Power simulation with coefficients with and without perturbation.

Chapter 6

Conclusion and Future Research

Several different methods to reduce power consumption in the multiplication component of the convolution of an FIR filter with a signal have been investigated. These methods fell into two classes of power reduction techniques. The first class used block processing to reorder the multiplications inherent in such a convolution. These methods involved the use of additional memory elements but did not change the result of the convolution in any manner. The second class sought to preprocess the FIR filter to reduce transitions within the coefficients of the filter. This preprocessing stage altered the coefficients and thereby changed the result of the convolution. However, any reductions of power would be gained without the cost of additional memory.

Two major block processing techniques were used. The first type was vertical unrolling. In this method, the filter coefficient was kept constant as an input to the multiplier through several multiplications. As additional unrollings were used, power was seen to go down monotonically. Overall, the sign-magnitude multiplier performed the best. Between two's complement multipliers, the Baugh-Wooley multiplier consumed less power than the Booth-Encoding multiplier. The second type of block processing was diagonal unrolling. In this method, the signal instead was kept constant for several multiplications. Power was reduced significantly with two unrollings, but increased with additional unrolling. Diagonal unrolling of order two kept at least one input to the multiplier constant between successive multiplications, thus accounting for this behavior. Once again, the sign-magnitude multiplier consumed the least power. In this case, the Baugh-Wooley multiplier performed significantly better than the Booth-Encoding multiplier. Comparison of the two methods

showed that vertical unrolling was a more effective method of reducing power.

The idea of unrolling could be applied to any set of multiplications with some sort of structure. In illustration, a simple experiment was performed with a Toeplitz matrix with and without consideration of the symmetry of the matrix. It was seen that significant reduction in power was realized when the symmetry was taken into account.

The second class of power reduction techniques involved coefficient perturbation. The method developed had two parts. The first part uniformly scaled the data to minimize transition activity within a range of allowable gain. The second stage iteratively considered the low order bits of the coefficients and set sets or subsets of these bits equal. Any modification to the coefficients was only accepted if the desired filter constraints were maintained. This method reduced transition activity within the coefficients by up to 56%. A power simulation of the largest reduction showed a reduction in power of 25% when the modified coefficients were used, with the filter characteristics maintained and no additional memory necessary.

There are several avenues for additional research in reducing power consumption in FIR filters. Using block processing, various other structures could be considered. The symmetry of most FIR windows could be taken advantage of to keep inputs to the multiplier constant for more multiplications. Additional research could also be performed in the second class of power reduction techniques. Many of the techniques that have been developed have been ad hoc. The theory behind such techniques could be investigated. The algorithm created in this thesis could be viewed as adaptive quantization. The general effect of such quantization on filters could be investigated. Finally, the idea of adaptive quantization could be incorporated with the Parks-McClellan algorithm to develop a self-contained algorithm that creates an optimal FIR filter with the additional constraint of minimizing transitions.

From the results in the two parts of this thesis – unrolling and perturbation – it is seen that there is a dependence of power on transition activity within multipliers when used in FIR filter convolution. If the costs of memory or slight alterations of results can be tolerated, power consumption can be reduced significantly by reducing transition activity, as seen in this thesis. There exists the opportunity for additional research in applying such techniques to make circuits more efficient in their computations.

Appendix A

Multipliers

In the course of running simulations to estimate power consumption, three multipliers were used. For data in sign-magnitude form, an array multiplier was used. For data in two's complement form, two multipliers were used – a Booth-encoding multiplier and a Baugh-Wooley multiplier.

A.1 Sign-Magnitude Array Multiplier

Sign-magnitude array multiplication is based on the premise that multiplication is the sum of several parallel products that can be computed in parallel. Two unsigned binary integers, X and Y, have values,

$$X = \sum_{i=0}^{m-1} X_i 2^i \tag{A.1}$$

$$Y = \sum_{j=0}^{n-1} Y_j 2^j, \tag{A.2}$$

where X_i and Y_j are the ith and jth bit values of X and Y, respectively. The product of these two values, P, is

$$P = X \times Y = (\sum_{i=0}^{m-1} X_i 2^i)(\sum_{j=0}^{n-1} Y_j 2^j) \tag{A.3}$$

$$= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (X_i Y_j) 2^{i+j} \tag{A.4}$$

$$= \sum_{k=0}^{m+n-1} P_k 2^k. \tag{A.5}$$

The terms, P_k, are called summands. For a given value of k, several partial products can be summed together. In Table A.1, the specific terms and their placement is shown for a 4×4 multipiler. From this presentation it is apparent that each bit of the product is a sum of terms that AND a bit of both the multiplicand, X and the multiplier, Y.

				X_3	X_2	X_1	X_0	Multiplicand
				Y_3	Y_2	Y_1	Y_0	Multiplier
				X_3Y_0	X_2Y_0	X_1Y_0	X_0Y_0	
			X_3Y_1	X_2Y_1	X_1Y_1	X_0Y_1		
		X_3Y_2	X_2Y_2	X_1Y_2	X_0Y_2			
	X_3Y_3	X_2Y_3	X_1Y_3	X_0Y_3				
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	Product

Table A.1: 4×4 Multiplier Partial Products

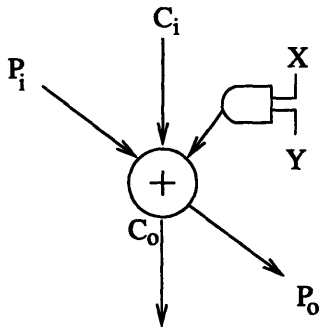

Figure A-1: Building block for sign-magnitude array multiplier.

The basic structure of the partial products allows the multiplier to be built from building blocks of the form depicted in Figure A-1. The input, P_i, is the previous partial product in the chain. The output, P_o is the partial product output of the current block. The carry input, C_i is the carry output from the next lower chain, while the carry output, C_o is the carry output of the current block. Finally, the inputs to the AND, X and Y, are single bits of the multiplicand and multiplier, repectively.

This building block can be used to construct a multiplier. A 4×4 version is shown in Figure A-2. [10] The sign-bit of the product is computed through combinational logic

acting upon the sign-bits of X and Y:

$$P_{sb} = X_{sb}\overline{Y_{sb}} + \overline{X_{sb}}Y_{sb}. \tag{A.6}$$

In the simulations, a larger 16 × 16 multiplier was used.

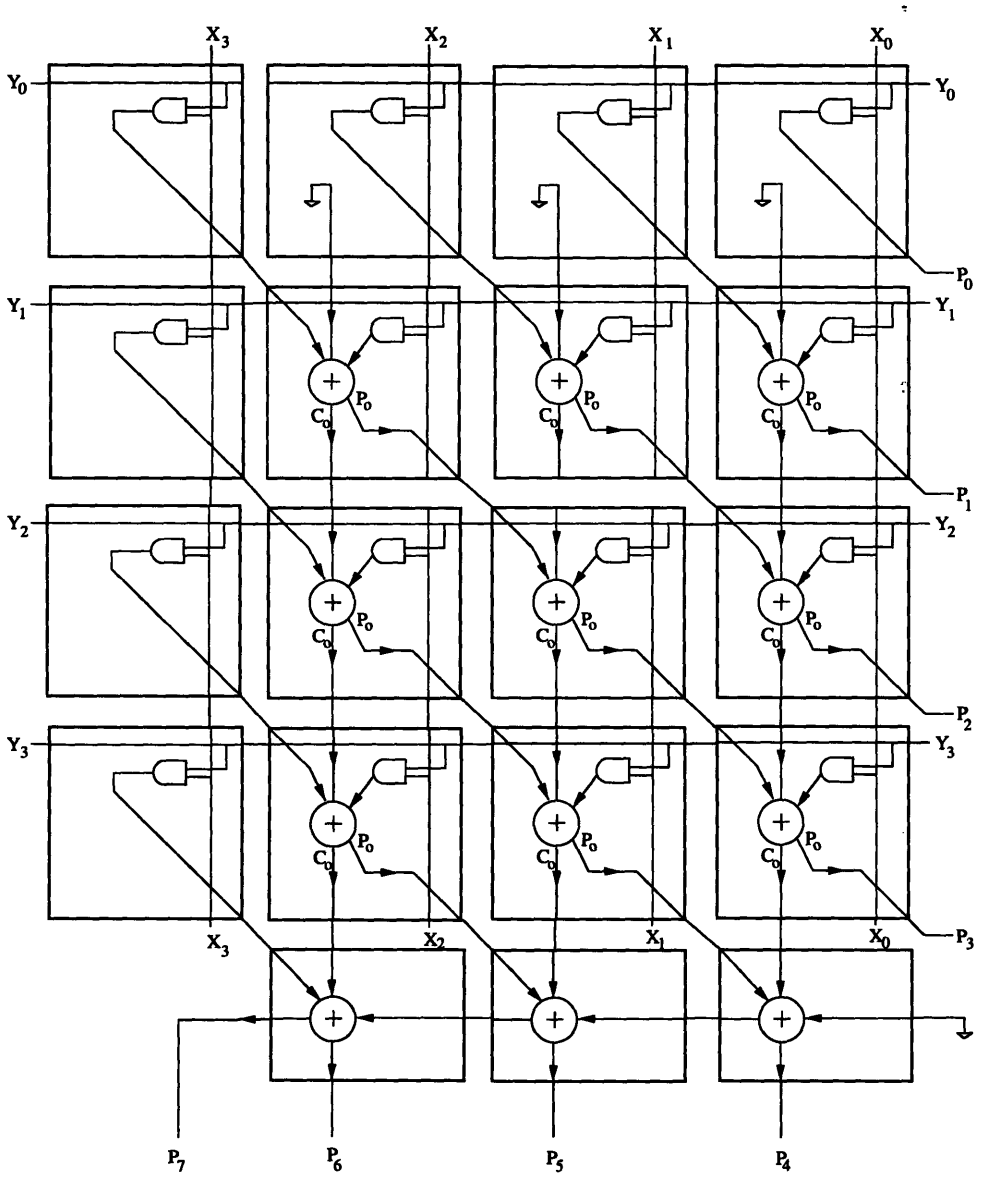

Figure A-2: 4 × 4 Sign-Magnitude array multiplier.

A.2 Baugh-Wooley Multiplier

Data is often represented in the two's complement form. In two's complement, multiplication is more difficult because the sign of a number is embedded in the number itself. Thus, the sign cannot be separated from the number as is the case in sign magnitude form, making multiplication more complicated. One multiplier that has been developed to handle two's complement inputs is the Baugh-Wooley multiplier.

In the multiplication to be performed, X is the m-bit multiplicand and Y is the n-bit multiplier. In two's complement form, these numbers are respesented in binary form by

$$X = -x_{m-1}2^{m-1} + \sum_{i=0}^{m-2} x_i 2^i \tag{A.7}$$

$$Y = -y_{n-1}2^{n-1} + \sum_{i=0}^{n-2} y_i 2^i, \tag{A.8}$$

where x_i and y_j are the ith and jth bits of X and Y, respectively. The value of the product of X and Y, P is

$$P = -p_{m+n-1}2^{m+n-1} + \sum^{m+n-2} p_i 2^i = XY \tag{A.9}$$

$$= (-x_{m-1}2^{m-1} + \sum_{i=0}^{m-2} x_i 2^i)(-y_{n-1}2^{n-1} + \sum_{i=0}^{n-2} y_i 2^i) \tag{A.10}$$

$$= (y_{n-1}x_{m-1}2^{m+n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{m-2} y_i x_j 2^{i+j}) \tag{A.11}$$

$$- (\sum_{i=0}^{m-2} y_{n-1}x_i 2^{n-1+i} + \sum_{i=0}^{n-2} x_{m-1}y_i 2^{m-1+i}). \tag{A.12}$$

In calculating P by summing the partial products, the sign of the products must be considered, particularly in terms containing x_{m-1} and y_{n-1}. These partial products can thus be reorganzed and written out in Table A.2. The first $n-2$ lines are added while the last two lines are subtracted.

The process of both adding and subtracting partial products can be cumbersome. However, instead of substracting partial products, the negation of these partial products can be added. For example, the negation of a k-bit two's complement number, Z, is

$$-Z = 1 - \bar{z}_{k-1}2^{k-1} + \sum_{i=0}^{k-2} \bar{z}_i 2^i, \tag{A.13}$$

					x_{m-1}	$\cdots$	x_3	x_2	x_1	x_0
						y_{n-1}	$\cdots$	y_2	y_1	y_0
						$x_{m-2}y_0$	$\cdots$	x_2y_0	x_1y_0	x_0y_0
					$x_{m-2}y_1$	$\cdots$	x_2y_1	x_1y_1	x_0y_1	
				$x_{m-2}y_2$	$\cdots$	x_2y_2	x_1y_2	x_0y_2		
				$\vdots$				$\vdots$		
	$x_{m-1}y_{n-1}$	0	$x_{m-2}y_{n-2}$	$\cdots$	x_2y_{n-2}	x_1y_{n-2}	x_0y_{n-2}			
0	0	$x_{m-2}y_{n-1}$	$x_{m-3}y_{n-1}$	$\cdots$	x_1y_{n-1}	x_0y_{n-1}				
0	0	$x_{m-1}y_{n-2}$	$x_{m-1}y_{n-3}$	$\cdots$	$x_{m-1}y_0$					
p_{n+m-1}	p_{n+m-2}	p_{n+m-3}	$\cdots$	p_{m-1}	$\cdots$	p_{n-1}	$\cdots$	p_2	p_1	p_0

Table A.2: General two's complement multiplication

where $\bar{z}_i$ is the binary complement of z_i. Therefore, the subtraction of

$$2^{n-1}(-0 \cdot 2^m + 0 \cdot 2^{m-1} + \sum_{i=0}^{m-2} y_{n-1}x_i 2^i) \tag{A.14}$$

can be replaced with the addition of

$$2^{n-1}(-1 \cdot 2^m + 1 \cdot 2^{m-1} + 1 + \sum_{i=0}^{m-2} \overline{y_{n-1}x_i} 2^i). \tag{A.15}$$

Thus, in the partial product represention of Table A.2, the line

$$0 \;\; 0 \;\; x_{m-2}y_{n-1} \;\; x_{m-3}y_{n-1} \;\; \cdots \;\; x_0y_{n-1} \tag{A.16}$$

is replaced by

$$1 \;\; 1 \;\; \overline{x_{m-2}y_{n-1}} \;\; \overline{x_{m-3}y_{n-1}} \;\; \cdots \;\; \overline{x_0y_{n-1}}, \tag{A.17}$$

with an additional "1" added in the p_{n-1} column. Similarly, the row

$$0 \;\; 0 \;\; x_{m-1}y_{n-2} \;\; x_{n-1}y_{n-3} \;\; \cdots \;\; x_{m-1}y_0 \tag{A.18}$$

is replaced with

$$1 \;\; 1 \;\; \overline{x_{m-1}y_{n-2}} \;\; \overline{x_{n-1}y_{n-3}} \;\; \cdots \;\; \overline{x_{m-1}y_0}, \tag{A.19}$$

with a "1 added in the p_{m-1} column. However, the introduction of Equation A.15 introduces a nonuniformity with regard to the partial product bits since in some cases a NAND is used

while in others a AND is needed. This can be simplified by noting the Equation A.15 has the value

$$\begin{cases} 0, & \text{for } y_{n-1} = 0 \\ 2^{n-1}(-2^m + 2^{m-1} + 1 + \sum_{i=0}^{m-2} \bar{x}_i 2^i), & \text{for } y_{n-1} = 1 \end{cases} \tag{A.20}$$

Therfore, Equation A.15 can be rewritten as

$$2^{n-1}(-2^m + 2^{m-1} + \bar{y}_{n-1}2^{m-1} + y_{n-1} + \sum_{i=0}^{m-2} y_{n-1}\bar{x}_i 2^i) \tag{A.21}$$

This equivalence and a similar one can be used to replace the last two rows of Table A.2 so that these rows can now be added. The resultant structure is described in Table A.3

					x_{m-1}	$\cdots$	x_3	x_2	x_1	x_0
						y_{n-1}	$\cdots$	y_2	y_1	y_0
						$x_{m-2}y_0$	$\cdots$	x_2y_0	x_1y_0	x_0y_0
					$x_{m-2}y_1$	$\cdots$	x_2y_1	x_1y_1	x_0y_1	
				$x_{m-2}y_2$	$\cdots$	x_2y_2	x_1y_2	x_0y_2		
				$\vdots$				$\vdots$		
	$x_{m-1}y_{n-1}$	0	$x_{m-2}y_{n-2}$	$\cdots$	x_2y_{n-2}	x_1y_{n-2}	x_0y_{n-2}			
	$\bar{y}_{n-1}$	$\bar{x}_{m-2}y_{n-1}$	$\bar{x}_{m-3}y_{n-1}$	$\cdots$	$\bar{x}_1y_{n-1}$	$\bar{x}_0y_{n-1}$				
	$\bar{x}_{m-1}$	$x_{m-1}\bar{y}_{n-2}$	$x_{m-1}\bar{y}_{n-3}$	$\cdots$	$x_{m-1}\bar{y}_0$					
1					x_{m-1}	y_{n-1}				
p_{n+m-1}	p_{n+m-2}	p_{n+m-3}	$\cdots$	$\cdots$	p_{m-1}	p_{n-1}	$\cdots$	p_2	p_1	p_0

Table A.3: General two's complement multiplication with all positive partial products

As only AND and ADD components are needed, the structure of the Baugh-Wooley multiplier is easily implemented by modifying the sign-magnitude array multiplier of Section A.1. In several partial products the complement of an input bit must be used. Also, the additional partial products, x_{m-1}, $\bar{x}_{m-1}$, y_{n-1}, $\bar{y}_{n-1}$, and "1", must be added. One fundamental difference between the sign-magnitude array multipier and the Baugh-Wooley multiplier is the independence of the sign bits in the first multiplier. Thus, a 5×5 sign-magnitude array multiplier will only have have a 4×4 array. In a Baugh-Wooley multiplier, on the other hand, the sign is implicit in the number. Thus, the same 5×5 multiplication will have a 5×5 array. This additional dimension results in Baugh-Wooley multipliers consuming more power than an equivalent sign-magnitude multiplier.[1]

A.3 Booth-Encoding Multiplier

Another multiplier that uses two's complement data as inputs is the Booth-Encoding multiplier. The concept for Booth-Encoding was developed by Andrew Booth. [2] Booth's technique was that multiplication could be performed irrespective of the sign of the inputs by using a method that observed several bits of the input at a time. Booth's algorithm was governed by a set of rules. In the multiplication of two numbers m and r, the nth digit of m (m_n) is examined:

(1)If $m_n = 0, m_{n+1} = 0$, multiply the existing sum of partial products by 2^{-1}, i.e. shift one place to the right.

(2)If $m_n = 0, m_{n+1} = 1$, add r into the eisting sum of partial products and multiply by 2^{-1}, i.e. shift one one place to the right.

(3)If $m_n = 1, m_{n+1} = 0$, subtract r from the existing sum of partial products and multiply by 2^{-1}, i.e. shift one place to the right.

(4)If $m_n = 1, m_{n+1} = 1$, multiply the sum of partial products by 2^{-1}, i.e. shift one place to the right.

(5)Do not multiply by 2^{-1} at m_0 in the above processes.

It is assummed that if m is given to n bits, then $m_{n+1} = 0$. Furthermore, all computations in the algorithm are performed modulus 2. An example multiplication using Booth's algorithm is shown in Figure A-3. A variation of Booth's algorithm is employed in the Booth-Encoding multiplier used in the simulations. This modified Booth algorithm uses a Radix-4 multiplication scheme two bits of the multiplier and a previous bit are analyzed at each stage. As the Booth-Encoding multiplier is quite common, no further detail shall be provided here. A complete description of the multiplier can be found in *Principles of CMOS VLSI Design* [10].

The nature of the Booth-Encoding multiplier is that several bits of the input are observed at one time. This makes it difficult for the multiplier to take advantage of single-bit correlation, as adjacent bits may vary independently of a given bit. This results in variation of the three-bit samples successive inputs. Accordingly, it can be expected that a Booth-Encoding multiplier will not benefit significantly from single-bit correlation in consecutive inputs.

	$m = 1.011(-\frac{5}{8})$	$r = 0.110(+\frac{3}{4})$	
$m_3 = 1, (m_4 = 0)$	subtract r	1.010	
	shift right		1.1010
$m_2 = 1, m_3 = 1$	shift right		1.11010
$m_1 = 0, m_2 = 1$	add r	1.11010	
		0.110	
		———	
		0.10010	
	shift right		0.010010
$m_0 = 1, m_1 = 0$	subtract r	0.010010	
		1.010	
		———	
		1.100010	
	no shift		1.100010$(= -\frac{15}{32})$

Figure A-3: A multiplication performed with Booth's algorithm