Power Reduction Techniques for a Spread Spectrum Based Correlator

David Garrett (garrett@virginia.edu) and Mircea Stan (mircea@virginia.edu)
Center for Semicustom Integrated Systems
University of Virginia - Department of Electrical Engineering
Charlottesville, VA 22903

Abstract

This paper presents the design of a low power spread spectrum correlator. We look at two major approaches and evaluate the best alternative for power reduction. We first consider a shift register FIFO implementation and look at reducing the switching activity for the arithmetic operations with a change in the addition algorithm. The correlation calculation can be modified to include storage of the previous result so that arithmetic circuits need only compute the difference between the present and next value. A binary adder tree with bypass can then reduce power by shutting off unnecessary computations. We then look at minimizing the power for sample storage by limiting the amount of data moved per cycle. This can be achieved by using a register file FIFO implementation. Interestingly, the two power minimization techniques, bypass adder tree and register file FIFO implementation, were found to be strongly non-orthogonal, with the final effect that the register file changes the data statistics in such a way that it cancels the savings for the adder tree with bypass. The final solution of a register file with standard adder tree was found to have the lowest power dissipation. Using Bus-Invert for encoding the data as it enters the FIFO further reduces the power consumption due to the global bus of the register file.

Keywords: *Direct sequence spread spectrum, adder tree with bypass, low power FIFO, Bus Invert.*

1. Introduction

In the design of a direct RF to baseband receiver, the phase shift keying (PSK) modulation with direct sequence spread spectrum (DSSS) requires despreading to recover the symbol data. A correlator is used to recognize certain spread spectrum signals and ignore others according to the despreading code. Figure 1 shows a block diagram of the receiver section, with the low noise amplifier (LNA), the quadrature mixer with local oscillator (LO), low-pass filters, A/D converters (ADC), and finally the despreading correlation block, which is the subject of this paper.

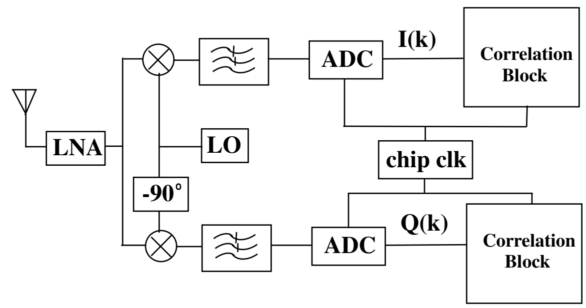


Figure 1: PSK DSSS Receiver Block Diagram

For mobile applications, the power consumption needs to be reduced to a minimum in order to maintain battery life. In the following sections we describe power reduction techniques by re-examining the structure of the correlator and then optimizing the arithmetic operations. Significant power savings in the correlator will have a large effect on the overall power consumed by the receiver.

2. Correlator Design

2.1 FIFO Sample Storage

After downconversion, the I and Q streams need to be sampled, digitized, and stored in a first-in-first-out (FIFO) for correlation. A typical implementation for a FIFO is an n-bit wide shift register of length 2^m-1 as shown in figure 2. As a sample comes in, another drops off of the end of the chain. All of the samples are passed into a correlation filter which performs sequence recognition on the samples according to internally stored (+1,-1) code coefficients.

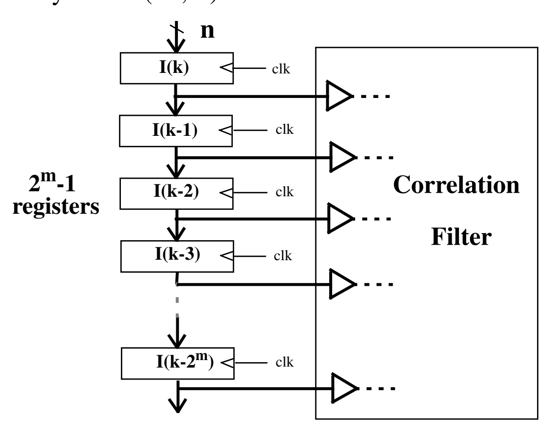


Figure 2: Shift Register FIFO implementation

The power consumed by CMOS circuits is directly proportional to the number of transitions, with a single transition dissipating $C_L V^2_{DD}$ [1]. The total switching energy can then be determined by summing all switching events:

$$E_{switching} = \sum C_L V_{DD}^2 \qquad (1)$$

In order to simplify the analysis, we will assume a constant logic voltage swing throughout the design, and consider that each gate provides a unit load to the corresponding driver. The power consumed by the shift register has two components:

- transitions on the register inputs and outputs,
- clock transitions.

One problem with the shift register implementation is that since the samples are moved every cycle, each register in the chain is switching all the time, unnecessarily consuming power. For random sample data, it can be shown that on average n/2 bits will transition per cycle at the output of each register as the data is passed through the FIFO [2]. The clock causes two extra transitions (rising and falling) per cycle on every flip-flop. Each register has an extra fanout load in the correlation filter. The power consumed in every cycle will then be proportional to the switching activity according to the following equations, where n is the number of bits in each sample, and 2^m-1 is the number of samples (number of coefficients in the DSSS code):

$$P_{SR} \propto bitshifts + clocktrans + fanout \qquad (2)$$

$$P_{SR} \propto \frac{n}{2}\left(2^m - 1\right) + 2n\left(2^m - 1\right) + \frac{n}{2}\left(2^m - 1\right) \qquad (3)$$

$$P_{SR} \propto 3n\left(2^m - 1\right)$$

A two register section of an 8-bit wide shift register chain was simulated using a transistor level model for the 2.0um Orbit process available through MOSIS. The power dissipation was estimated from simulation by integrating the supply current through an RC network [3]. For a 25MHz clock, and a 5Volt supply, the average power dissipation per 8-bit flip-flop for random data was 3.3mW. Each register sees the load of the next register and the buffers into the correlation adder tree. By extrapolation, the average power consumption for an entire 255 length shift register is approximately 842mW for a 5Volt supply. Lowering the supply voltage will decrease power consumption quadratically [1], hence this simulation data is only meant to be comparative for the switching activity, not the absolute minimum.

2.2 Adder Tree Design

The next major block of the correlator is the arithmetic core. The correlator is designed to despread incoming sequences based on a finite impulse response (FIR) filter equation with binary coefficients (+1 and -1). Figure 5 shows the standard block diagram for such a filter taking the samples from the registers (the control logic for the registers has been removed for simplicity). The adder network computes the sum which is compared to a threshold. When the incoming samples are aligned with the code coefficients, the correlation will have a large value. If the samples are not aligned, or if a different code was used at the transmitter, the sum will be much less then the threshold.

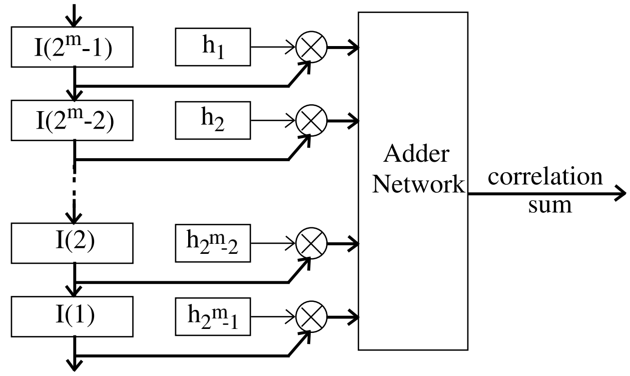


Figure 5: Correlator Block Diagram

The equations for the correlation filter can be described as:

$$CS(k) = \sum_{n=1}^{2^m - 1} h_n I(k-n) \qquad (4)$$

where k is the time step.

2.3 Maximum Length Codes

The despreading codes for our application of DSSS are maximum length sequences generated by linear feedback shift registers (LFSR) like the eight bit generator in figure 6, which generates a pseudorandom sequence of length 255.

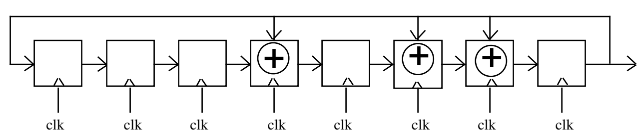


Figure 6: LFSR Code Generator

The code sequence has distinct properties that characterize the coefficients over the length of the code. In particular, the *run property* defines the number of runs (streams of consistent 1s or 0s) to be dependent solely on the length of the code [4]. The runs are as follows:

- one run of 1's of length *m*,
- one run of 0's of length *m-1*,
- one run of 1's and one run of 0's of length *m-2*,
- two runs of 1's and two runs of 0's of length *m-3*,
- four runs of 1's and four runs of 0's of length *m-4*,
- " "
- 2^{m-2} runs of 1's and 2^{m-2} runs of 0's of length *1*.

2.4 Algorithm Change for Low-Power Addition

The run property of maximum length codes allows us to reduce the number of transitions in the correlator design as only half of the terms in the correlation sum will have a different coefficient and change their contributions to the overall sum in each cycle. Although the data will have shifted one state, the previous coefficient and the new coefficient will remain the same for half the number of samples (in runs of length 2 or greater). In order to capture this behavior we define "bypass bits" which will be set for terms that are not changing (see figure 7). These status bits tell the adder stages if a term is not changing and if it has zero contribution to the difference between the present and next correlation sums.

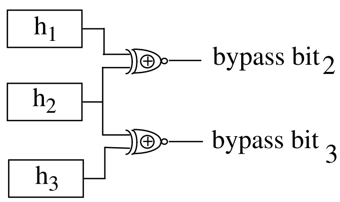


Figure 7: Bypass bit generation

By identifying the factors that have changed, and by storing the previous sum, we can streamline the arithmetic operation to reduce the number of terms. Although we cannot reduce the overall number of adders (in any one clock cycle, any adder can be required), we can shutdown the unused adders, and prevent power consumption. The equation for the correlator can be rewritten to express this new method as follows:

$$CS(k) = CS(k-1) + 2\sum_{n=1}^{2^m-1}\left(h_n^*\right)I(k-n) \qquad (5)$$

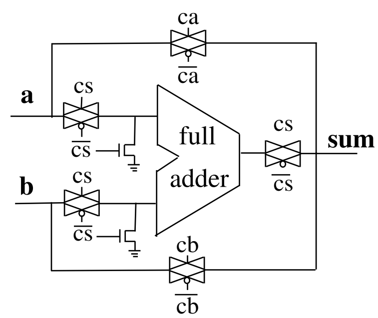
$$h_n^* = h_n\left(h_n \oplus h_{n-1}\right) \qquad (6)$$

If the coefficient for a sample has not changed from the previous calculation, then h^*_n is 0 in equation (6), otherwise h^*_n will reflect the new polarity (+1 or -1). When the coefficient changes, the original sample value must be removed from the sum, and then the sample with the new polarity must be added. This can be handled in one step by adding twice the sample with the new polarity (which explains the 2 before the summation symbol in equation 5).

Also, in each cycle, the newest sample that enters the chain must be added and the offgoing sample must be subtracted from the overall correlation sum.

2.5 Adder Tree with Bypass

In order to take advantage of the new method of calculating the correlation sum, a specialized adder cell was developed to take advantage of the properties of the maximum length code. In the case where a coefficient has not changed as a sample is shifted, its particular contribution is zero to the overall sum. When a term is bypassed, the adder can be configured to ignore its value, and only pass the other input as the result. Figure 8 shows a full-adder surrounded by passgates. According to the state table, when the bypass bit is set for the *a* input, the lower passgate allows *b* to pass along as the output. In this case the adder inputs are disconnected, and no changes are propagated to the internal adder circuitry. When both bypass bits are set, the adder cell is completely removed from the chain and the adder cell propagates a bypass status bit along with its output to the next stage in the binary adder tree.



Za Zb	ca cb cs
0 0	0 0 1
0 1	1 0 0
1 0	0 1 0
1 1	0 0 0

Figure 8: Modified Adder Cell with bypass

Let us consider the performance of the bypass adder as compared to the simple adder block. When both inputs are active, the bypass adder suffers from the overhead of the passgates. On average, an 8-bit adder cell suffers a 7% power dissipation increase as measured in SPICE simulations for random data. The bypass adder is burdened from an overhead in the case where it is adding two numbers, but it uses much less power in the other three cases (bypass modes). When either of the inputs is bypassed, or when the entire adder is shut down, the bypass adder cell power consumption becomes almost negligible.

Figure 9 shows a slice of the first layer of the correlation adder tree, and how the simple adder tree differs from the

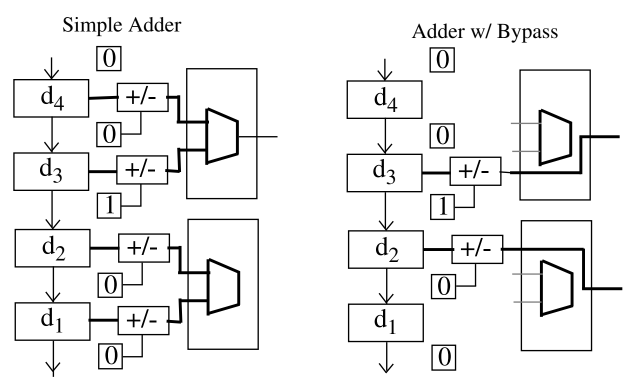


Figure 9: A hardware comparison of the simple adder the adder with bypass

adder with the bypass cells. In the simple adder case, regardless of the coefficient, the data is recomputed on every cycle in every adder. The adder with bypass on the other hand, only adds in computations when the successive coefficients are different. The passgates remove unused adders from the tree and allow single values to continue to the next level.

The run property statistics determine how many bypass bits are set and the particular spread spectrum code dictates where the bypass bits are located. For example, a run of three coefficients will set two bypass terms. Depending on how the bypass bits fall on the adder tree, either two adders will be in single bypass mode, or one adder will be in shutdown mode. With runs of four or more, at least one adder will be shutdown.

Using these observations and the average power dissipation values recorded from each of the bypass configurations, the overall power dissipation of the correlation adder tree can be accurately estimated. Table 1 shows the power dissipation calculations for the first three rows of the bypass adder configuration. The bypass adder configuration numbers were generated from a typical set of coefficients from the maximum sequence generator as seen in figure 6.

Rows of adder tree	bypass adder configurations	Pavg/cell (mW)	power (mW)
first row	33 - on 60 - bypass 35 - off	1.5	50
second row	36 - on 23 - bypass 5 - off	1.7	61
third row	32 - on	1.9	61

Table 1: Bypass Adder Power Dissipation

As the data passes through the adder tree, it becomes highly correlated but we can still estimate the overall power consumption of the binary tree using approximations based on the simulation data. Power consumption for the adders cells were taken from 8 bit adders. Each successive row adds another bit, so the power consumption for each adder can be computed by multiplying by the ratio of bits to the original 8-bit cell.

Table 2 shows power calculations for the standard and bypass adder tree implementations. According to the estimate, the bypass cells can reduce power consumption by 29% as compared to the regular correlation adder tree for 8-bit data samples (371mW versus 531mW).

The advantage of the bypass adder is that the unused adders are held in a latched state so that no transitions occur internally when data is bypassed. To fully realize the correlation filter, an additional 16 bit adder stage is required to add the previous sum to the current difference value stored in a latch. The latch power has been extrapolated from the shift register simulations. The final adder tree results must be shifted left by one place (multiply by two) before being added to the previous sum.

Components	Pavg/ cell (mW)	Pavg/ row (mW)	Pavg/cell (mW)	Pavg/ row (mW)
	Simple Adder		**Bypass Adder**	
128 (8 bit)	1.4	179	from table 1	50
64 (9 bit)	1.6	102	from table 1	61
32(10 bit)	1.8	58	from table 1	61
16 (11 bit)	1.9	30	1.9	30
8(12 bit)	2.1	17	2.1	17
4(13 bit)	2.3	9	2.3	9
2(14 bit)	2.5	5	2.5	5
1(15bit)	2.6	3	2.6	3
1(16bit)	n/a		2.8	3
16 latch	n/a		6.6	7
coefficient multiplier (256)	0.5	128	0.5	128
overall power estimate		**531**		**374**

Table 2: Adder Tree Power Consumption

3. Alternate Correlator Design

3.1 Register File Storage

In order to reduce the correlator switching activity, a different starting point is to use a register file (with pointer) FIFO implementation instead of the n-bit wide shift register, as seen in figure 10 [5]. With this scheme, only one register out of the total of 2^m-1 will experience clock and output transitions. The trade-off is that now a global bus must be

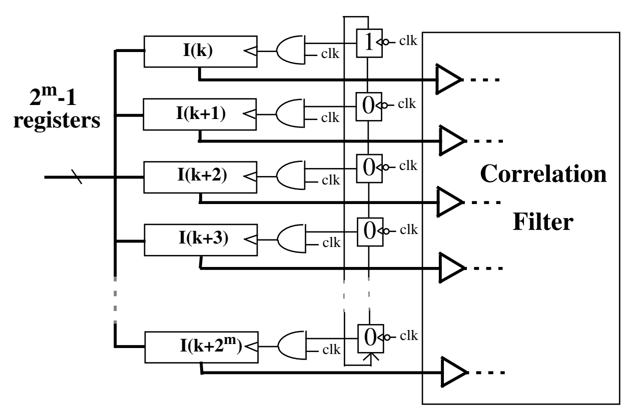


Figure 10: Register File Concept

connected to each register, increasing the load due to the inputs of all the registers. To create the illusion of the FIFO structure, we also need a one-hot shift register of length (2^m-1) as a pointer to the register to be loaded with the incoming sample.

The power consumed by the register file FIFO comes from seven main components:

- transitions on the global bus,
- the fanout of the registers into the correlation block,
- the clocked registers,
- the clock transitions on the AND gates,
- the clocks on the address bit registers,
- the hot-bit shifting through that register,
- the filter coefficients that must be now rotated.

The total number of transitions per cycle becomes:

$$P_{RF} \propto bus + fanout + regclk + gatedclk + bitclk + bitshift + coefclk + coefs \quad (7)$$

$$P_{RF} \propto \frac{n}{2}\left(2^m - 1\right) + \frac{n}{2} + 2n + 2\left(2^m - 1\right) + 2\left(2^m - 1\right) + 2 + 2\left(2^m - 1\right) + 128 \quad (8)$$

$$P_{RF} \propto \frac{n+12}{2}\left(2^m - 1\right) + \frac{5n+260}{2}$$

3.2 Register File versus Shift Register FIFO

When a comparison of the power between the register file and the shift register is plotted for varying *n* and *m*, it is shown that the register file has a power advantage as long as the bus width is greater than four bits and there are more than 31 samples. For any practical sample storage requirement, the register file has an immediate impact on the power dissipation. Figure 11 shows a plot of the power reduction graph over various samples sizes and bus widths. The practical power savings tracks an asymptote for maximum power reduction for each of the particular bus widths.

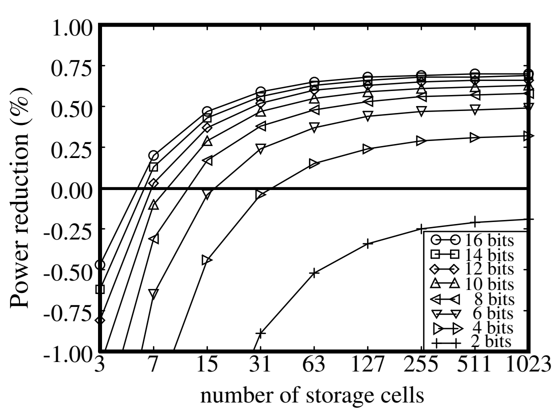


Figure 11: Power Reduction of Register File over Shift Register

In order to find an absolute power consumption approximation for the register file, the SPICE simulation results can be used in conjunction with theoretical equations developed in this paper. For an 8-bit register file with a 255 length code, the average power consumption at 25MHz is approximately 56% of the shift register implementation (842mW as computed in section 2.1) and the register file will have an average power consumption of around 370mW.

3.3 Bus Invert

A decomposition of the power equation for the register file shows that the majority of the power is consumed in the global bus that drives the registers (in the case of the shift register, the global bus does not exist). A proven technique for reducing power consumption on a global bus is the Bus Invert method [6]. In this technique, an extra line is added to the bus to encode the data to have fewer transitions between samples. If the Hamming distance between the current sample and the previous sample is greater then n/2, the data is inverted to provide a closer match to the current bus state, and the invert bit is used to store the inversion state. On an 8 bit communication bus, this can be shown to reduce the transitions by approximately 20%, and can be easily used for the register file implementation of the FIFO for power reduction. In our application, the overhead for Bus Invert is reduced because anyhow the samples need to be available in either the inverted or non-inverted form depending on the correlation coefficients. Adjusting the global bus factors with a 20% derating from equation 8 yields the relative power consumption for the register file with Bus Invert as follows:

$$P_{RF} \propto \frac{0.8n+12}{2}\left(2^m - 1\right) + \frac{4.8n+260}{2} \quad (9)$$

For an 8-bit wide 255 length register file FIFO, Bus Invert lowers the overall power consumption by an additional 7.5% for a total reduction in 60% over the shift register.

3.4 Arithmetic Operations with Register File FIFO

By optimizing the FIFO implementation to use a register file, the input statistics to the arithmetic unit are significantly changed. The data is now mostly static with the coefficients shifting around them. This leads to the unexpected outcome that using the bypass adder tree with this configuration actually adds power overhead to the correlation calculation. Whereas the bypass adder significantly reduced power for the shift register FIFO due to the changing inputs, the register file FIFO inherently keeps adder inputs stable even without the bypass cells. A simulation of a single adder with stable sample data and the polarity changing according to the DSSS code coefficients had an average power dissipation of 0.8mW (compare with 1.4mW for shift register and changing samples as in table 2) in the first row while a bypass adder in the first row of the adder tree had a power dissipation of 1.7mW. Similarly, simulations on the second row for the regular adder tree showed an average power dissipation of 1mW per adder cell (compare with 1.6mW in table 2). The bypass adder had an average power dissipation of 1.9mW in the second.

Simulation data was not available for the remaining rows of the register file FIFO adder tree because of the computational expense. Since each third row adder inputs depend on four adders from the first row, and the probability of all eight of the inputs to the first row remaining static was very low, the power consumption on the third row and lower was estimated as if the data was randomly changing. Recomputing the adder tree power consumption as in table 2 results in a power estimate of 416mW for the regular adder tree fed by the register file FIFO. Recomputing the bypass adder tree calculations with the new data gives a power estimate of 594 mW.

4. Overall Comparison

When all of the alternatives are evaluated, the clear winner is the register file FIFO with Bus Invert coupled with a regular adder tree. As seen in table 3, it has a power consumption of only 54% of the shift register FIFO with regular adders and 60% of the shift register FIFO with bypass adders. The bypass adder tree reduced power when inputs were changing into the adder tree (29% power reduction in the adder tree, 11% for the overall correlator) power reduction, but the register file FIFO not only lowers the power consumption in the sample storage circuitry, but also considerably reduces the switching activity into the adder tree. The FIFO optimization and the adder tree with bypass turn out to be two non-orthogonal greedy algorithms where minimizing each of the individual components does not lead to the global optimum.

Design	storage Pavg (mW)	adder tree Pavg (mW)	total Pavg (mW)	nor-malized
Shift register regular adders	842	531	1373	1.0
Shift register bypass adders	842	374	1216	0.89
register file regular adders	370	416	786	0.57
register file bypass adders	370	594	964	0.70
register file w/ bus invert	337	416	753	**0.54**

Table 3: A Comparison of Power Saving for Correlation Implementations

5. Conclusions and Future Work

We have presented several power minimization techniques for a direct sequence spread spectrum correlator working at the chip rate. Depending on the FIFO implementation (shift register or register file), different adder tree solutions are optimal for low power design. When samples are shifted each cycle (as for the shift register FIFO), an adder tree with bypass reduces the overall power by 11%. When the samples are static and only the coefficients are shifted (as for the register file FIFO), a regular adder tree gives the best results for an overall 43% power reduction. Using Bus Invert further reduces the overall power by an extra 3%.

Future work should include a VLSI implementation of the low power correlator followed by actual power measurements in order to verify simulations and analytical results.

Acknowledgments

The authors would like to thank Dr. Jim Harris for inspiring some of this work and Dr. Ron Williams, Dr. Steve Jones, Max Salinas, Adam Von Ancken, and Peter Schaefer for many interesting discussions. This work was partially supported by NSF Career Grant MIP-9703440.

References

[1] A. Chandrakasan, I. Yang, C. Vieri, D. Antoniadis, *Design Considerations and Tools for Low-voltage Digital System Design*, Proceedings of the Design Automation Conference, pp. 113-118, June 1996.

[2] M. Stan, W. Burleson, *Low-Power Encodings for Global Communication in CMOS VLSI*, to appear in IEEE Transactions on VLSI Systems, 1997.

[3] S. Kang, *Accurate Simulation of Power Dissipation in VLSI Circuits*, IEEE Journal of Solid-State Circuits, Vol SC-21, No. 5, October 1986, p. 899-901.

[4] R. Ziemer, R. Peterson, *Digital Communications and Spread Spectrum Systems*, Macmillan Publishing Company, 1985, pp. 386.

[5] E. Tsern, T. Meng, *A Low Power Video-Rate Pyramid VQ Decoder*, IEEE Journal of Solid-State Circuits, November 1996.

[6] M. Stan, W. Burleson, *Bus-Invert Coding for Low Power I/O*, IEEE Transactions on VLSI Systems, March 1995, p. 49-58.