Performance Analysis of Error Control Codes for Wireless Sensor Networks

Gopinath Balakrishnan[†], Mei Yang[‡], Yingtao Jiang[‡], and Yoohwan Kim[*]
[†] Texas Instruments Inc., Dallas, TX 75243
[‡] Department of Electrical and Computer Engineering
[*] School of Computer Science
University of Nevada, Las Vegas, NV 89154
Emails: [†]gopi@ti.com, [‡]{ meiyang, yingtao}@egr.unlv.edu, [*]yoohwan@cs.unlv.edu

Abstract

In wireless sensor networks, the data transmitted from the sensor nodes are vulnerable to corruption by errors induced by noisy channels and other factors. Hence it is necessary to provide a proper error control scheme to reduce the bit error rate (BER). Due to the stringent energy constraint in sensor networks, it is vital to use energy efficient error control scheme. In this paper, we focus our study on the performance analysis of various error control codes in terms of their BER performance and power consumption on different platforms. In detail, error control codes with different constraints are implemented and simulated using VHDL. Implementation on FPGA and ASIC design is carried out and the energy consumption is measured. The error control performance of these codes is evaluated in terms of Bit Error Rate (BER) by transmitting randomly generated data through a Gaussian channel. Based on the study and comparison of the three different error control codes, we identify that binary-BCH codes with ASIC implementation are best suitable for wireless sensor networks.

Keywords - *Wireless sensor network; Error Control Code; BER; power consumption.*

1. Introduction

The low-cost, rapid deployment, ability of self-organization and cooperative data-processing, have made wireless sensor networks a practical solution for a wide range of application areas, including military and homeland security, health, environment, industry and commercial, and home [1]. The most significant challenge in sensor networks is to overcome the energy constraints since each sensor node has limited energy to consume. Since data transmitted over the wireless media is vulnerable to corruption by noise, error control schemes are necessary to keep the Bit Error Rate (BER) low. Due to the stringent energy constraint, it is impossible to increase the signal power of the transmitted signal in wireless sensor networks. Hence an alternative way is to use the error control codes to reduce the BER. The encoding and decoding circuitry for error control codes may consume a sizable amount of power. This motivates us to study energy-efficient error detection/correction codes.

In the literature, there is limited work on energy-efficient error control schemes for sensor networks and some of them are reviewed as follows. In [1], it is shown that usage of ARQ is limited for sensor networks due to the additional retransmission energy cost and overhead. In [8], convolutional codes are analyzed for power in a frequency nonselective, slow Rayleigh fading channel. Results show that the energy required for encoding data is negligible. However, performing Viterbi decoding on a Strong-ARM processor using a C compiler is energy-intensive as the average energy consumption per useful bit grows exponentially with the constraint length of the code and independent of code rate. Using forward error correction (FEC) is inefficient if the decoding is performed using a microprocessor, for which a dedicated onboard Viterbi decoder is suggested [8]. To the best of our knowledge, study of other error control codes for sensor networks is not available in the literature.

In [7], the Minimum Energy (ME) coding scheme for sources with unknown statistics and a new method of code-by-code detection that can detect and correct certain errors in the received codeword is proposed. This research combines the modulation with the error control scheme to minimize power consumption. The on/off key modulation performance is improved with a ME-Coding. However, the ME codes have no capability of error correction [7].

To identify energy-efficient error control codes for sensor networks, in this paper, we study and analyze the performance of several error control codes. Power consumption in a circuit is directly proportional to its complexity. In general, the encoder consumes negligible power when compared to the decoder. Thus, the challenge is to choose an error control code with less complex decoding circuit.

In our study, we consider two types of codes: linear block codes and convolutional codes. Linear block codes can be either cyclic or non-cyclic. Cyclic codes are of interest and importance due to their rich algebraic structure which has extremely concise specifications and can be efficiently implemented using simple shift registers [5]. The most widely used cyclic codes for wireless applications are BCH codes [4]. For the purpose of our study, binary-BCH codes and the most popular non-binary BCH codes, namely Reed Solomon (RS) codes, are studied and analyzed. For comparison purpose, we also studied and analyzed the convolutional code with Viterbi algorithm.

The rest of the paper is organized as follows. Section 2 discusses the methodology used to conduct the performance analysis. Section 3 presents and compares the BER performance and power consumption of several error correction codes and identifies the energy-efficient error control code with the results obtained.

2. Methodology

In this work, we evaluate the power consumption of three different FEC codes, BCH, RS, and convolution codes on different platforms. The implementation on general processors may be inefficient due to the limitation of the compiler and other factors [8]. Hence, we implement the three codes with different constraints using hardware description language and estimate their power consumption on FPGA and ASIC. The code with the least power consumption is identified. All the comparison is based on the assumption of the same error control performance which is evaluated by the BER test. In the following, we explain the methods used in our study.

2.1. Implementation of Codes

The three types of error correction codes are implemented using VHDL. Fig. 1 illustrates the procedure of encoding and decoding in a communication system, where u is the information word, v is the codeword, v' is the received word and u' is the decoded word. The encoder circuits of linear block codes and convolutional codes have simple hardware and are easy to implement. Some of the issues considered while implementing the decoder circuits are as follows.

- In binary-BCH and RS codes, the Euclid's Algorithm (EA) [9] the Berlekamp-Massey Algorithm (BMA) [1][5] can be used to compute the coefficients of error polynomial $\sigma(x) = \sigma_0 + \sigma_1 x + ... + \sigma_t x^t$. In EA, all the steps used for computation are identical and easy to implement in hardware. Hence for efficient hardware implementation EA is preferred than BMA [6].

- The information received by the receiver shown in Fig. 1 is quantized before decoding. Depending on the level of quantization, the decoding can be classified into Hard Decision Decoding (HDD) or Soft Decision Decoding (SDD) [6]. The HDD is used when the quantization level is two while SDD is used for quantization level greater than two. The SDD performs better than HDD but requires highly complex circuitry [6]. Hence HDD is implemented to minimize the power in the decoder [6].

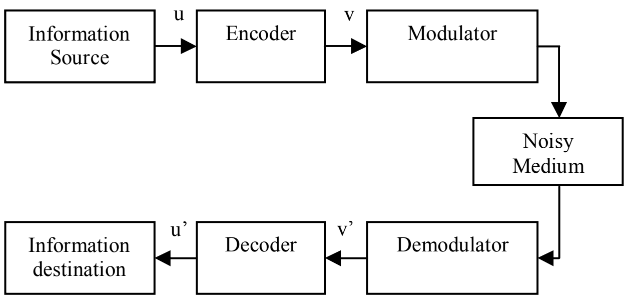

Figure 1. Procedure of encoding and decoding in a communication system.

2.2. Performance Measure

The next step is to measure the error correcting capability of the implemented codes which is given by BER, which is obtained by the number of erroneous bits divided by the total number of transmitted bits. BER is affected by several factors including noise in the channel, quantization technique used, code rate R, energy per symbol to noise ratio E_s/N_o and transmitter power level P_{out}. The *code rate* is given by $R = k/n$, where k is the number of bits at the input of the encoder and n is the number of bits at the output of the encoder. The BER is shown to be directly proportional to the code rate and inversely proportional to energy per symbol noise ratio and transmitter power level [2].

The encoder encodes the data with code rate R and transmits it over the noisy channel. If the transmitter power level P_{out} is unchanged, then the received energy per symbol E decreases to $R*E$. Hence, the BER measured at the input of the decoder is larger than the BER of the data transmitted without coding [2]. This increase in BER is overcome by using a decoder that can correct errors. Proper choice of error correction codes will reduce the BER to several orders of magnitude. The difference in BER achieved by using error correction codes to that of uncoded transmission is referred to as *coding gain*. The BER test is performed by simulations on Matlab following the procedure shown in Fig. 1.

First the information bits are generated using a random number generator. The randomly generated data is then sent to the encoder circuit and encoded into code words, which are transmitted over the noisy channel. Before transmitting, the encoded data is modulated using Phase Shift Keying (BPSK), which is done by mapping 1/0 at the output of the encoder to -1/+1 of an antipodal baseband signal [3].

To evaluate the performance of the error control codes in the noisy channel, an Additive White Gaussian Noise (AWGN) channel is modeled. Adding Gaussian noise to the encoded data is done by generating Gaussian random numbers with desired energy per symbol to noise ratio. The variance σ^2 of additive Gaussian noise which has the power spectrum of $N_o/2$ Watts/Hz is equal to $N_o/2$. If the energy per symbol E_s is set to 1, then we have $E_s/N_o = 1/2\ \sigma^2$ and the standard deviation σ is given by $\sigma = sqrt(1/2(E_s/N_o))$. Hence, the standard deviation σ with desired E_s/N_o is calculated and used to obtain a Rayleigh random variable R as shown in Eq. (1).

$$R = sqrt(2*\sigma^2*ln(1/1-U)), \qquad (1)$$

where σ^2 is the variance of the Rayleigh random variable and U is a uniformly distributed random number.

The Gaussian random number G obtained by using Rayleigh random variable R is given by Eq. (2).

$$G = \mu+R*cos(2*\pi*T), \qquad (2)$$

where T is a uniformly distributed random number and μ is the mean of the Gaussian random variable. The generated Gaussian noise is then added to the encoded bits and transmitted.

The received symbols are quantized and fed to the decoder to obtain the information bits. Quantization refers to the process of approximating the continuous set of values with a finite set of values. As explained in Section 2.1, the 2-level quantization is used to reduce the complexity of the decoder. In the 2-level quantization, the received signal is mapped to 0 if the signal level is greater than zero and mapped to one if the signal level is less than 0. The result obtained in this way is called hard decision. The hard decision decoder is used to decode the quantized data which are 1's and 0's.

The decoded data is then compared with the corresponding input given to the encoder and the BER is calculated. The BER of the uncoded channel is theoretically calculated using Eq. (3).

$$P(e) = \tfrac{1}{2} *erfc(sqrt(E_b/N_o) = Q(sqrt(2E_b/N_o)). \qquad (3)$$

The performance of the coded and uncoded channels is compared based on the calculated BER.

2.3. Power Estimation in FPGA Design

The FPGA used for this work is Xilinx Virtex-E XCV200E-6CS144 [12]. The Xilinx Integrated Software Environment (ISE) 7 along with Xilinx Power (XPower) [12] is used for estimating power. XPower calculates the power in the design by summing up the power consumed by each element. The power consumed by each switching element in the design is given by Eq. (4).

$$P = C*V^2*E*F, \qquad (4)$$

where P represents the power in mW, C represents the capacitance in Farads, V represents the voltage in Volts, E represents the switching activity (average number of transitions per clock cycle), and F represents the frequency in Hz.

2.4. Power Estimation in ASIC Design

Power estimation in ASIC is studied using Synopsys's Design Compiler (DC) and Design Vision [10]. The power analysis in ASIC is performed in the following procedure. First the VHDL design is analyzed to check if it uses the synthesizable VHDL subset. Then the design is elaborated, where the design is built with generic and technology-independent components like Gates, Flip Flops, MUX, etc. It is followed by uniquify, where multiple copies of the sub-design are made whenever it is referred in the upper level of the hierarchy, and each copy is optimized in a unique way according to the conditions and constraints. The last step of synthesis is compiling, where the network generic components is translated into a netlist of the target library. Compilation can be constrained in terms of power. The power report is then generated.

3. Performance Analysis

In this section, we present and compare the performance of the binary-BCH, RS codes, and Viterbi codes in terms of their error correction capability in bit error rate (BER) and complexity and power consumption on FPGA and ASIC.

3.1. BER Test

Fig. 2 compares the BER of uncoded channel, binary-BCH and RS codes. A (n, k, t) binary-BCH and RS code can correct up to t-errors, where t is directly proportional to the number of parity bits (n-k) [4]. Hence the performance of various binary-BCH and RS codes are analyzed by varying n and k and their corresponding energy consumption is measured. For this purpose, implementation is done by varying *(n-k)*.

For binary-BCH codes, n is chosen as 31 and k varies in {26, 21, 16, 11}. It is clear from Fig. 2 that the BCH(31,11,5) code, which can correct up to 5 errors has the lowest BER curve with the highest coding gain of 5 dB and the lowest code rate of 0.355. While the BCH(31,26,1) code, which can correct up to 1 error has very minimal coding gain with the highest code rate of 0.839. The BCH(31,16,3) and BCH(31,21,2) have coding gain of 4 dB and 2 dB with code rate of 0.516 and 0.677, respectively.

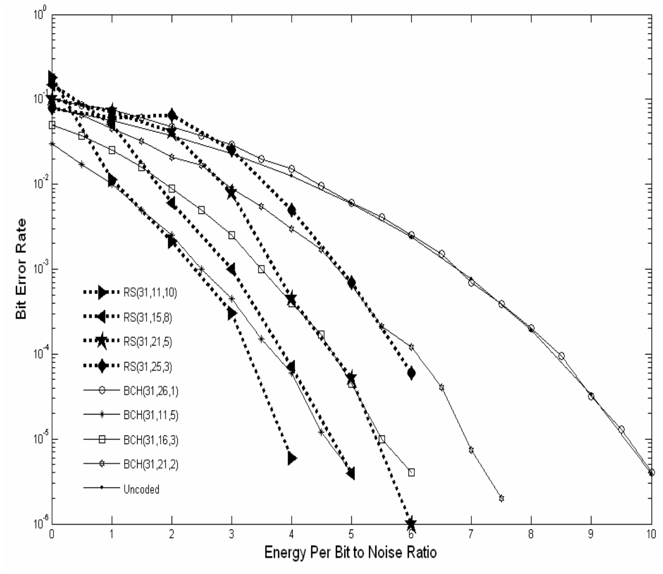

Figure 2. BER of linear cyclic block codes.

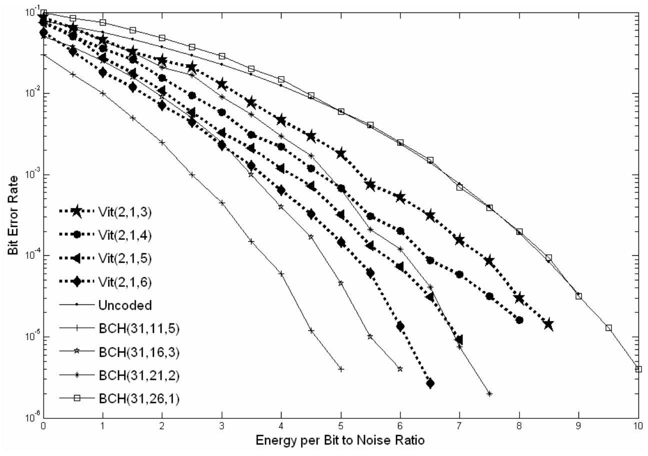

Figure 3. BER of convolutional codes.

For RS codes, n is chosen as 31 and k varies in {25, 21, 15, 11}. Fig. 2 clearly shows that the coding gains of RS codes are not the same for all the BER. They perform worse than the uncoded channel for higher values of BER and provide asymptotically high gain at lower values of BER. From the figure, we can see that using RS codes for BER of 10^{-1} or higher results in channel loss while using them for BER lesser than 10^{-1} results in extremely high gain. Because the RS codes correct symbol errors, they are more suitable for applications with BER lesser than 10^{-1}. For BER lower than 10^{-1}, the RS(31,11,10) code, which can correct up to 10 symbol error, has the lowest BER curve with a coding gain of approximately 6 dB and a minimum code rate of 0.355. While the RS(31,25,3) code capable of correcting 3 symbol errors has the largest BER curve with a coding gain of 2 dB and a maximum code rate of 0.806. The RS (31,15,8) code and the RS (31,21,5) code have coding gain of 4 dB and 3 dB and code rate of 0.4838 and 0.677, respectively.

Fig. 2 also compares the binary-BCH codes with the RS codes based on code rates. The binary-BCH codes show a better performance at higher BER than the RS codes. But for lower BER, the RS codes perform better than the binary-BCH codes. For example, considering BCH(31,16,3) and RS(31,15,8) with code rate 0.5, for BER greater than 10^{-2}, the BCH code performs better than the RS code, while for BER less than 10^{-2} the RS codes perform better.

Unlike block codes, low BER and large coding gain in (n, k, m) convolutional codes are achieved not by increasing k and n but by increasing the memory order m. Hence to analyze the performance of these codes, various convolutional codes of different memory orders are implemented and their energy consumption is measured. In this thesis, simulation is done for convolutional encoders with fixed code rate R = ½ and by varying memory order m in {3, 4, 5, 6}.

Fig. 3 shows that the coding gain increases with the increased memory order m. The Viterbi (VIT) (2,1,3), (2,1,4), (2,1,5), and (2,1,6) codes approximately have a coding gain of 1 dB, 2 dB, 3 dB and 4 dB, respectively. Convolutional codes implemented using soft decision decoder improves the coding gain by 3 dB than that implemented using hard decision decoder [3]. The tradeoff is that soft decision decoder requires highly complex hardware [6].

3.2. Complexity and Power Analysis in FPGA

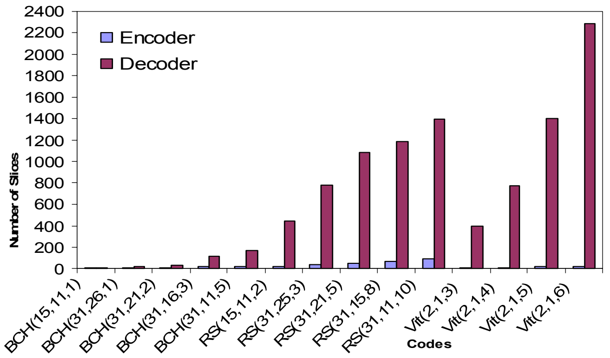

Figure 4. Complexities of BCH, RS, and Viterbi Codes.

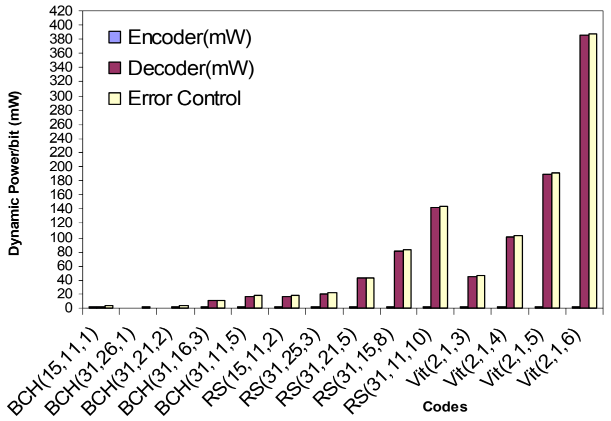

Figure 5. Power consumption in FPGA design.

The complexity of the FPGA implementation is measured by the number of slices used on Xilinx XCV200E. The slice utilization of each of the codes obtained from the synthesis report is plotted in Fig. 4. From the figure, it can be inferred that the complexity of the encoder and decoder circuitry of BCH codes and RS codes increases linearly with the number of parity bits (n-k) increasing. On the other hand, the complexity of the Viterbi decoder circuitry increases alarmingly with memory order m increasing.

It can be observed that the encoder circuit of the convolutional codes and the binary-BCH codes are less complex than that of the RS codes. This is because the RS codes use non-binary encoders. For decoder, the binary-BCH codes have less complex circuit than those of the RS codes and the convolutional codes. Power estimation of all the codes using XPower is plotted in Fig. 5. Among all the codes, the binary-BCH codes consume the least power.

3.3. Power Analysis in ASIC

Power consumption in ASIC design is obtained as explained in Section 2. The target library used is *lsi_10k.db* [10]. Fig. 6 shows the power consumed by the encoder and decoder circuitry of all three type of codes in ASIC. It is clear that the power consumption in binary-BCH and RS codes are directly proportional to the number of parity bits and the power consumption of the convolutional code is proportional to the memory order *m*. Also the power consumed by the decoder circuit is significantly higher than that consumed by the encoder circuit of all the codes. The decoding of RS codes and convolutional codes consume significantly larger amount of power compared to that of the binary-BCH codes. And the binary-BCH codes consume the least power among all the codes.

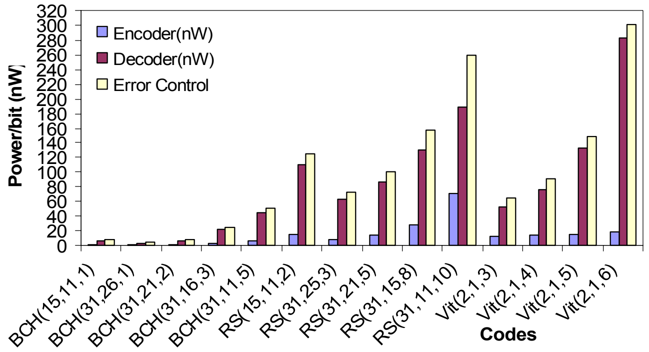

Figure 6. Power consumption in ASIC design.

3.4. Power Analysis for Sensor Nodes

In the following, we analyze the power consumption of the encoding/decoding circuit in a sensor node. As the power consumption in ASIC implementation is much less than that in FPGA implementation, we use the power results in ASIC implementation for the analyze based on *TSMC 0.18μm* by taking 2.5 times the results obtained using *lsi_10k.db* (as we measured through experiments). The total power consumed in the sensor node for communication with coded channel is given by Eq. (5) [8].

$$E(k,d) = \{E_T(k,d)+E_{encode}\}+\{E_R(k)+E_{decode}\}, \qquad (5)$$

where E_T is the energy used by the transmitter circuitry, which is a function of the no. of message bits (k) and the distance between the sensor nodes (d); E_R is the energy used by the receiver circuitry, which is given by $E_{elec}*k$; E_{encode} and E_{decode} is the energy used to encode and decode respectively, which can be obtained from the results in Section 2.2.

For uncoded channel, the receiving power is same as that of the coded channel, while the transmission power is given by Eq. (6).

$$E_T(k,d)=E_{elec}*k + E_{amp}*k*d^2, \qquad (6)$$

where E_{elec} is the energy consumed by transmitter/receiver circuit; E_{amp} is the energy consumed by the amplifier; k is the no. of message bits; d is the distance between the sensor nodes. For our analysis d^2 is chosen as 500, k = 1201200, E_{elec} = 50*nJ/bit* and E_{amp} = 100*pJ/bit/m*^{2}.

Fig. 7 compares the power consumption of the coded channel and uncoded channel with encoders/decoders implemented in ASIC. It can be inferred that the power consumed by the binary-BCH codes and RS codes for coded channel is less than that of the uncoded channel. While coded channel using convolutional encoder with Viterbi decoder consumes more power than that of the uncoded channel. Hence it is clear that convolutional codes are not suitable for wireless sensor networks.

Fig. 8 clearly shows that the significant amount of power consumed by the sensor node goes for error control if convolutional codes are used while power consumed by transceivers dominate the total power consumption if binary-BCH codes or RS codes are used. Hence, linear cyclic block codes consume significantly less power than uncoded channel.

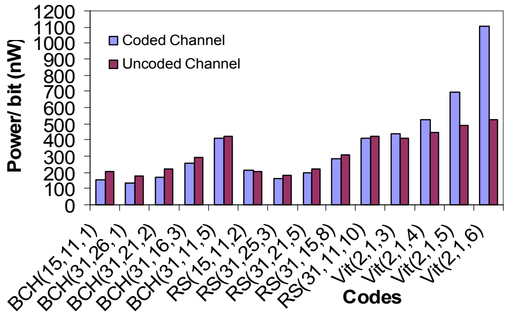

Figure 7. Power consumption in sensor node for coded and uncoded Channels.

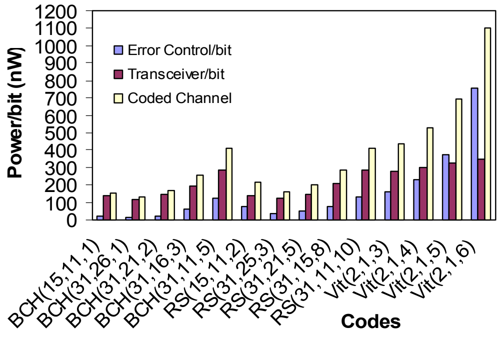

Figure 8. Power consumption in transceivers and error control circuitry.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Communications Magazine*, pp. 102-114, Aug. 2002.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer Networks: The Int'l Journal of Comp. and Telecom. Net.*, vol. 38, no.4, pp.393-422, Mar. 2002.

[3] Chip Fleming, *A Tutorial on Convolutional Coding with Viterbi Decoding*, Spectrum applications, Derwood, USA, 1999.

[4] S. Lin and D.J. Castello, *Error control coding, Fundamentals and applications*, Premtice-Hall, New Jersey, 1983.

[5] J.L. Massey, “Shift-register synthesis and BCH decoding”, *IEEE Trans. Inf. Theory*, vol. 15, pp. 122-127, January 1969.

[6] Robert H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, John Wiley & sons, New York, 2002.

[7] Y. Prakash and S.K.S. Gupta, “Energy efficient source coding and modulation for wireless applications,” *IEEE Trans. Inform. Theory*, pp. 212-217, Mar. 2003.

[8] E. Shih *et al.*, “Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks,” *Proc. ACM MobiCom '01*, Rome, Italy, pp. 272–86, Jul. 2001.

[9] Y. Sugiyama, M. Kasahara, S.Hirasawa, and T. Namekawa, “A method for solving key equation for decoding Goppa codes,” *Inf and Control*, vol. 27, pp. 87-99, 1975.

[10] Synopsys Inc., http://www.synopsys.com/products/logic/design_compiler.html.

[11] TSMC Inc., http://www.mosis.org/products/fab/vendors/tsmc/tsmc018.

[12] XILINX, The Programmable Logic Data Book, Xilinx Inc, San Jose, CA, 2002.