International Journal of Future Generation Communication and Networking
Vol.10, No.6 (2017), pp.55-64
http://dx.doi.org/10.14257/ijfgcn.2017.10.6.06

Square Code in WSNs (Wireless Sensor Networks)

Joong-Ho Lee

Yongin University, Computer Science Dept., Korea
Joongho65@yongin.ac.kr

Abstract

This paper proposed the Square code which is appropriate to the WSNs. The Square code can detect failures that are larger than double bits error and can correct failures that are larger than single bit error, and the implementing method is presented in the paper. For an information that are 16 bits long, all kinds of single bit and double bits error can be detected and corrected. For the triple bit errors, all of the errors can be detected and 93.6% can be corrected where the triple bit error combination has 560ea(C(16,3)). Proposed code scheme is suitable in the WSNs because it has a small size packet. Also, this paper classifies a fault model in the sensor node. Proposed Square code has improved area-overhead more than 50% compared with SEC-DED(Single Error Correcting – Double Error Detecting) conventional codes.

Keywords*: WSNs, Square code, error detect, error correct, single bit error, double bit error, triple bit error, area-overhead, SEC-DED*

1. Introduction

The development of semiconductor technology has led to the scaling down of feature sizes and lowering of operating voltage [1]. Because of those reasons, WSNs have emerged from the convergence of smart sensor technology, wireless communication, and processing technology. Especially, scaling down is effective for the low power operation because it allows sensor nodes to become smaller in size and be more efficient in power usage [2].

WSNs might be growing rapidly. Also, it will be employed in a wide range of new applications including various environment monitoring, such as earthquake, forest fire, battlefield surveillance, machine failure diagnosis, biological detection, home security, smart spaces and inventory tracking in the near future [3,4]. WSNs are deployed randomly to a wide area which is unreachable from human activity area. So, they are exposed weakly to a hostile environment because they have a mission to monitor natural environment under power constraints and severe weather conditions. Even though under hostile environment, wireless sensor networks need reliable data collection and transmission to meet the mission of monitoring. Additionally, maintaining reliability of the sensor networks from the hardware failure is very important, because there is very little human intervention to repair or maintain the sensor network systems since deployed [5,6].

As the sensor node becomes smaller, the required operating power also decreases. This effectively reduces the cost for wireless sensor networked systems deployment. When it comes to reliability, the smaller and cheaper wireless sensor networks are harder to gather and transmit to a robust data. So, the reliability factor of the system becomes more important when a wireless sensor network system becomes larger. Additionally, even if the sensor node has a power constraint, the wireless sensor networked system has to guarantee to provide reliable data. Also, wireless communication channel can also be another source of failure when transmitting data from other sensor node [7].

WSNs are consisted of tiny sensor nodes that are able to communicate each other

ISSN: 2233-7857

wirelessly to transmit collected data [5]. A set of sensor nodes build clusters, and one particular sensor node which is elected among the sensor nodes in the cluster becomes a cluster head. In order to transmit data, data is transferred to the cluster head and then transferred to other cluster. Finally, data is transferred to the sink node which is the final station of the transmitted data. As mentioned before, if a hardware failure occurs in the sensor node, the sensor node must be able to recover automatically because human can't repair the fault in sensor node since deployed. So, we need to apply the fault recovery technology which can detect error and correct error bits in WSN.

This paper proposed a reliability improvement methodology in the wireless sensor networks. First, this paper analyzes failure mechanism in wireless sensor networks. Second, new error correcting code is proposed which can correct single error data bit and detect larger than 2 bits error in the wireless sensor network based on the analyzed failure mechanism.

2. Error Control in Wireless Sensor Networks

We need to focus on a cost effective way to control malfunction in WSN systems. ARQ (Automatic Repeat Request) is one of the strategies to control errors in WSN. As retransmit the error packet, it can control error when the system found errors. These error packets are retransmitted until there is error free. In this case, system can detect error by using a cyclic redundancy check (CRC). To detect error from the transmitted data, it needs additional error check bits among the original data. This automatic repeat request (ARQ) scheme has increased cost from resending error packet when it occurs error [8,9].

ATM-8 HEC code is commonly used in the CRC. This code scheme checks the data bit error for the 64 bits data that are suitable to the semiconductor memory system. To detect data bits error for the 64 bits, 7 CRC parity bits are added to detect data bit corruption, and ATM-8 HEC code generates the error correcting code from the polynomial x^8+x^2+x+1.

$$\text{code word } (\boldsymbol{n}) = \text{information bits}(\boldsymbol{k}) + \text{CRC parity bits}(\boldsymbol{p}) \quad (1)$$

A code word of length $\boldsymbol{n}$ is consisted of the information bits length $\boldsymbol{k}$ together with the parity bits of length $\boldsymbol{p}$. For the implementation of the ATM-8 HEC code by using the polynomial, the code is made of 6 XOR gate logic layers, and in total it has the overhead of over 700 XOR gate. [10]

FEC (Forward Error Correction), which can improve the performance of error control, is another strategy for controlling the error in the WSN. It is based on the error correcting codes (ECCs) that can correct received error bits. Data that are sent to the receiver include ECCs which is able to detect and correct error bits using certain amount of redundant information. Therefore, FEC can reduce power consumption in the process because they don't need to retransmit error packets [11,12].

Data packet size is increased by the redundancy information, which is consisted of parity bits. Thus, even without the redundancy costs calculated, it needs additional costs for the encoding in the transmitter and the decoding in the receiver. Many types of ECCs were introduced in the previous research. Hamming code is the most typical code in the ECC. And Reed-Solomon and BCH codes are widely known block codes. The ECC block codes are commonly represented by the triple $(\boldsymbol{n, k, p})$, code rate is defined as $\boldsymbol{Rc = k/n}$.

Assume that the data consists of a certain number of information bits $\boldsymbol{k}$. A number of check bits $\boldsymbol{p}$ identifies the error which may be one of the check bits. The number of check bits $\boldsymbol{p}$ can distinguish $2^{\boldsymbol{p}}$ cases. Therefore, check bits must check themselves as well as the information bits, the value of $\boldsymbol{p}$, must range from 0 to $p+k$, which is $p+k+1$ distinct value.

$$2^{\boldsymbol{p}} \geq p+k+1 \quad (2)$$

It applies to any single error correcting (SEC) binary FEC block code.

3. Error Mechanism in Wireless Sensor Networks

There are many source of data corruption when the signal is propagated through the wireless channel, such as noise and interference. Data error can be investigated and classified in the hardware side between the transmitter and receiver.

3.1. Fault Model

WSNs have many clusters and sensor nodes and a sink node. Regarding failures, it is necessary to approach the different parts for each functional model, such as between cluster and cluster, between sensor node and cluster or between cluster node and sink node. Figure 1 shows the failure in the WSNs.

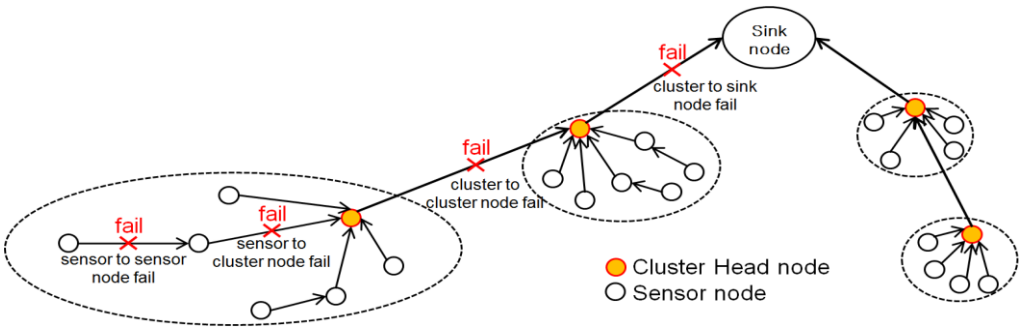

Figure 1. Fault Model in WSNs

3.1.1. Sensor Node

- Sensing fail: It is depending on the quality of sensor. It can be excluded from the classification of the malfunction.
- Stuck-at fault: “1/0” fixed fault in the input/output stage.
- Transition fault: The failure that occurs when a data is transition from “0/1” to “1/0”.

3.1.2. Transmitter and Receiver

This is a type of failure that occurs between the sensor node and CH (Cluster Head) node or between the sensor nodes. It can be classified into following fault:

- Stuck-at fault: “1/0” fixed fault in the input/output stage.
- Transition fault: The failure that occurs when a data is transition from “0/1” to “1/0”.

These malfunction occur between the transmitter and receiver due to external noise such as alpha particle.

4. The Square Code

4.1. Information Length $k = 4$

This paper proposed a Square code which is based on the Matrix type CRC code. [13] The proposed coding scheme essentially checks for the bit errors in the row direction and column direction by using the parity bits. In the Table 1, it showed the configuration of the code word, where n (= 8) is the length of a code word, k (= 4) is the number of information bits in a code word and p (= 4) is the number of parity bits. This code represented as (8, 4, 4). Parity bits are located in the end of the column and row direction as shown in Table 1. Based on Table 1, code word is encoded at the first transmitting terminal, when it is received from sensor node to the other sensor node. According to syndrome inequality, it is decoded at each sensor node to determine whether there are errors or not. Syndrome inequality (*s0~s3*) is shown in Table 1.

Data is transferred to the receiver sequentially when the transmitter sends data to the receiver. Figure 2 shows data sequence which is the code word of 8 bits length and this code word is consists of 4 bits information and 4 bit parity in a clocked system.

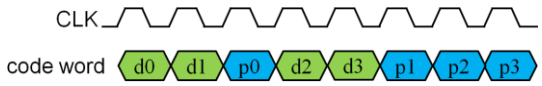

Figure 2. Data Sequence in Clock System

Therefore, code word is received to the receiver as following sequence:

$$\text{code word } (n) = d0\ d1\ p0\ d2\ d3\ p1\ p2\ p3 \tag{3}$$

Parity *p0* checks the odd numbers bit of '1' among the *d0* and *d1*. Parity bits *p0~p1* check the data bits in a row direction, and parity bits *p2~p3* check the data bits in a column direction. In order to organize odd-parity check system, if the number of '1' data is odd, the parity bit will be a '1'. If not, the parity bit will be a '0'. Finally, Syndrome *s0* must be '0' that is the sum of *d0*, *d1* and *p0*. Due to an error, if the number of '1' bits are odd, syndrome will be '1'. All of the syndrome values (*s0~s3*) are calculated in the same way and the syndrome expression are shown in Table 1. And (8, 4, 4) code is shown in Table 2. Minimum distance of this code has d_{min} = 3. Therefore, it is possible to distinguish the 16 cases of information.

Table 1. Configuration for the 4 Bits Information

	4 bits		
	data	data	parity
data	d0	d1	p0
data	d2	d3	p1
parity	p2	p3	

S0 = d0 ⊕ d1 ⊕ p0
S1 = d2 ⊕ d3 ⊕ p1
S2 = d0 ⊕ d2 ⊕ p2
S3 = d1 ⊕ d3 ⊕ p3

Table 2. (8, 4, 4) Square Code

d0	d1	p0	d2	d3	p1	p2	p3
0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1
0	0	0	1	0	1	1	0
0	0	0	1	1	0	1	1
0	1	1	0	0	0	0	1
0	1	1	0	1	1	0	0
0	1	1	1	0	1	1	1
0	1	1	1	1	0	1	0

d0	d1	p0	d2	d3	p1	p2	p3
1	0	1	0	0	0	1	0
1	0	1	0	1	1	1	1
1	0	1	1	0	1	0	0
1	0	1	1	1	0	0	1
1	1	0	0	0	0	1	1
1	1	0	0	1	1	1	0
1	1	0	1	0	1	0	1
1	1	0	1	1	0	0	0

4.1.1. Coverage

(8, 4, 4) code has 4 parity bits. There is minimum distance 3 between the each code word. All kinds of single bit and double bits error can be detected and corrected. Triple bits error can be detected but can't be corrected. From Table 1, quadruple bits error can't be detected because all of syndrome values have logic low status.

4.2. Information Length *k* = 8

Based on the minimum Hamming distance, the length of parity bits should be increased proportionally for the extending information bits. In Table 2 it is shown the (14, 8, 6) code configuration of the code word, where *n* (= 14) is the length of a code word, *k* (= 8) is the number of information bits in a code word and *p* (= 6) is the number of parity bits. Syndrome expression is consisted of the same way in Table 1, and shown in Table 3. Code word is received to the receiver as following sequence:

$$\text{code word } (n) = d0\ d1\ d2\ p0\ d3\ d4\ d5\ p1\ d6\ d7\ p2\ p3\ p4\ p5 \tag{4}$$

Table 3. Configuration for the 8 Bits Information

	8 bits			
	data	data	data	parity
data	d0	d1	d2	p0
data	d3	d4	d5	p1
data	d6	d7	-	p2
parity	p3	p4	p5	

S0 = d0 ⊕ d1 ⊕ d2 ⊕ p0
S1 = d3 ⊕ d4 ⊕ d5 ⊕ p1
S2 = d6 ⊕ d7 ⊕ d8 ⊕ p2
S3 = d0 ⊕ d3 ⊕ d6 ⊕ p3
S4 = d1 ⊕ d4 ⊕ d7 ⊕ p4
S5 = d2 ⊕ d5 ⊕ d8 ⊕ p5

4.2.1. Coverage

(14, 8, 6) code has 6 parity bits. There is minimum distance 6 between the each code word. All kinds of single bit and double bits error can be detected and corrected. For the triple bit errors, all of the errors can be detected and 82.2% can be corrected where the triple bit error combination has 56ea(C(8,3)). 7% of quadruple bits errors can't be detected (36 patterns) for example *d0/d1/d3/d4* bits error is occurred because all of syndrome values have logic low status.

4.3. Information Length *k* = 16

When the information length is extended to 16 bits, code is organized to (24, 16, 8). In Table 4 it is shown the configuration of the code word, where *n* (= 24) is the length of a code word, *k* (= 16) is the number of information bits in a code word and *p* (= 8) is the number of parity bits. Code word is received to the receiver as following sequence:

code word (*n*) = *d0 d1 d2 d3 p0 d4 d5 d6 d7 p1 d8 d9 d10 d11 p2 d12 d13 d14 d15 p3 p4 p5 p6 p7* (5)

Table 4. Configuration for the 16 Bits Information

	16 bits				
	data	data	data	data	parity
data	d0	d1	d2	d3	p0
data	d4	d5	d6	d7	p1
data	d8	d9	d10	d11	p2
data	d12	d13	d14	d15	p3
parity	p4	p5	p6	p7	

S0 = d0 ⊕ d1 ⊕ d2 ⊕ d3 ⊕ p0
S1 = d4 ⊕ d5 ⊕ d6 ⊕ d7 ⊕ p1
S2 = d8 ⊕ d9 ⊕ d10 ⊕ d11 ⊕ p2
S3 = d12⊕ d13 ⊕ d14 ⊕ d15 ⊕ p3
S4 = d0 ⊕ d4 ⊕ d8 ⊕ d12 ⊕ p4
S5 = d1 ⊕ d5 ⊕ d9 ⊕ d13 ⊕ p5
S6 = d2 ⊕ d6 ⊕ d10 ⊕ d14 ⊕ p6
S7 = d3 ⊕ d7 ⊕ d11 ⊕ d15 ⊕ p7

4.3.1. Coverage

(24, 16, 8) code has 8 parity bits. There is minimum distance 8 between the each code word. All kinds of single bit and double bits error can be detected and corrected. For the triple bit errors, all of the errors can be detected and 93.6% can be corrected where the triple bit error combination has 560ea (C(16,3)). 4% of quadruple bits errors can't be detected (72 patterns) for example *d0/d1/d4/d5* bits error is occurred because all of syndrome values have logic low status.

4.4. Information Length *k* = 32

In order to increase information length to 32 bits, code must be organized to (44, 32, 12). In Table 5 it is shown the configuration of the code word, where *n* (= 44) is the length of a code word, *k* (= 32) is the number of information bits in a code word and *p* (= 12) is the number of parity bits. Code word is received to the receiver as following sequence:

code word (*n*) = *d0 d1 d2 d3 d4 d5 p0 d6 d7 d8 d9 d10 d11 p1 d12 d13 d14 d15 d16 d17 p2 d18 d19 d20 d21d22 d23 p3 d24 d25d26 d27 p4 d28 d29 d30 d31 p5 p6 p7 p8 p9 p10 p11* (6)

Table 5. Configuration for the 32 Bits Information

Pin	32 bits						
	data	data	data	data	data	data	parity
data	d0	d1	d2	d3	d4	d5	p0
data	d6	d7	d8	d9	d10	d11	p1
data	d12	d13	d14	d15	d16	d17	p2
data	d18	d19	d20	d21	d22	d23	p3
data	d24	d25	d26	d27			p4
data	d28	d29	d30	d31			p5
parity	p6	p7	p8	p9	p10	p11	

S0 = d0 ⊕ d1 ⊕ d2 ⊕ d3 ⊕ d4 ⊕ d5 ⊕ p0
S1 = d6 ⊕ d7 ⊕ d8 ⊕ d9 ⊕ d10 ⊕ d11 ⊕ p1
S2 = d12 ⊕ d13 ⊕ d14 ⊕ d15 ⊕ d16 ⊕ d17 ⊕ p2
S3 = d18 ⊕ d19 ⊕ d20 ⊕ d21 ⊕ d22 ⊕ d23 ⊕ p3
S4 = d24 ⊕ d25 ⊕ d26 ⊕ d27 ⊕ p4
S5 = d28 ⊕ d29 ⊕ d30 ⊕ d31 ⊕ p5
S6 = d0 ⊕ d6 ⊕ d12 ⊕ d13 ⊕ d14 ⊕ d15 ⊕ p6
S7 = d1 ⊕ d7 ⊕ d13 ⊕ d19 ⊕ d25 ⊕ d29 ⊕ p7
S8 = d2 ⊕ d8 ⊕ d14 ⊕ d20 ⊕ d26 ⊕ d30 ⊕ p8
S9 = d3 ⊕ d9 ⊕ d15 ⊕ d21 ⊕ d27 ⊕ d31 ⊕ p9
S10 = d4 ⊕ d10 ⊕ d16 ⊕ d22 ⊕ p10
S11 = d5 ⊕ d11 ⊕ d17 ⊕ d23 ⊕ p11

Table 6 shows minimal solutions of SEC-DED (Single Error Correcting - Double Error Detecting) code which is based on inequality (1) for a range of values of information bits ***k***. Proposed Square code is shown in Table 1 which is compared with Hamming code. For example, a total 24 bits length of code word required eight check bits including 16 information bits and Hamming code needs five bits for the same code word length that is to provide the SEC-DED.

4.4.1. Coverage

(44, 32, 12) code has 12 parity bits. There is minimum distance 12 between the each code word. All kinds of single bit and double bits error can be detected and corrected. For the triple bit errors, all of the errors can be detected and 97.7% can be corrected where the triple bit error combination has 4960ea (C(32,3)). 2% of quadruple bits errors can't be detected for example *d0/d1/d6/d7* bits error is occurred because all of syndrome values have logic low status.

Table 6. Parity Bits Comparison for Error Correction/Detection

Number of Information Bits	Number of Parity Bits for SEC-DED	
	Hamming Code	Square Code
1	2	2
2	3	3
3~4	3	4
5~6	4	5
7~9	4	6
10~11	4	7
12	5	7
13~16	5	8
17~20	5	9
21~25	5	10
26	5	11
27~30	6	11
31~36	6	12
37~42	6	13
43~49	6	14
50~56	6	15

5. Square Code Implementation

5.1. Error Correction Block

The block which is correcting the error is implementation for the proposed Square (24, 16, 8) code is shown in the Figure 3. According to the syndrome inequality in Table 4, bit errors can detect. For example, if data *d0* is corrupted, syndrome value *S0/S2* will get logic high, and then *S0/S2* will detect error in *d0*. In this case, the gate AD turns on the

N1/P1 transistor, and then the corrupted data *d0* will go through a transfer gate N1/P1. Finally, corrupted data *d0* is corrected to the original value, which has a complementary logic value of *d0*. If there is no error, the gate ND turns on the N2/P2 transistor, and then the original data d0 will go through a transfer gate N2/P2. All of the single bit error among the data (*d0~d15*), it can be corrected as the same way.

For the double-bit errors, there are 120 cases error patterns where C(16,2) is the combinations of double bit error. And 24 case error patterns (20%) can't be correctable. For example, if double bits errors occur in the data *d0/d1*, syndrome value *S4* and *S5* will get logic high but *S0* still stay logic low states. Therefore, according to the *S0/S4*, data *d0* can't be corrected also *d1* can't be corrected from the syndrome *S0/S5*.

And only some of the odd-bit errors for the data, it can be corrected. For example, if triple error occurs in the data *d0~d2*, syndrome value *S0*, *S4*, *S5*, and *S6* will get logic high. Therefore, according to the *S0/S4*, data *d0* can be corrected, and *d1* can be corrected from the syndrome *S0/S5*, and *d2* can be corrected from *S0/S6*.

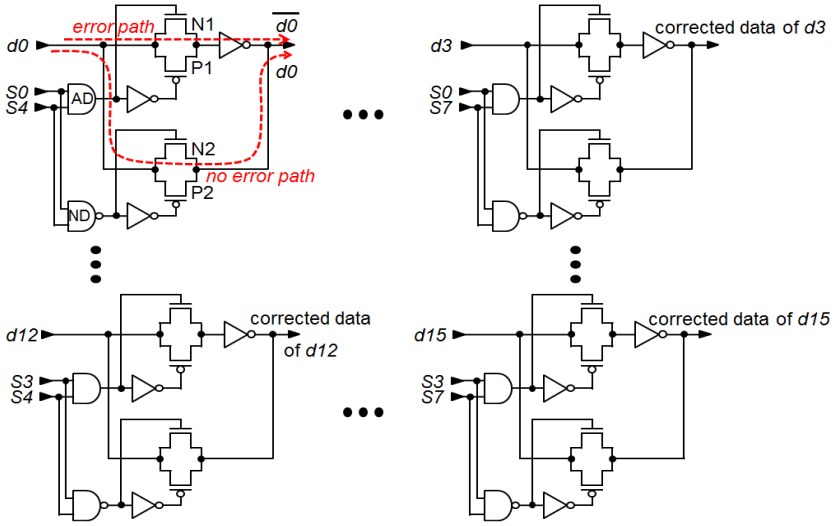

Figure 3. Error Correction Block for the Proposed Square (24, 16, 8) Code

5.2. Error Detection Block

Figure 4 shows the implementation of decoding block for the proposed Square (24, 16, 8) code and Figure 5 shows the error detecting block. For example, if double bits errors occur in the data *d0/d1*, syndrome value *S4* and *S5* will get logic high but *S0* still stay logic low states. All of the double bit errors can be detected by compare with *S3* and *S4* or compare with *S5* and *S6*.

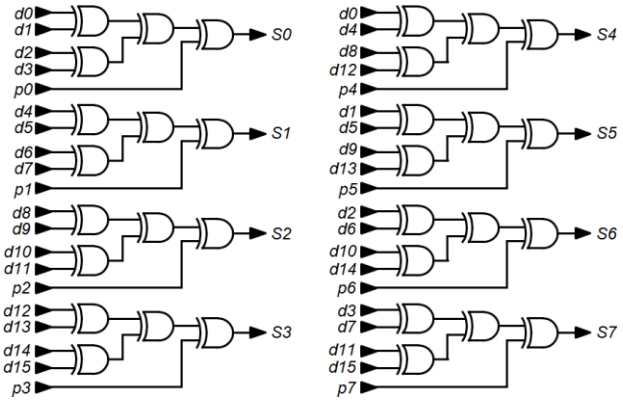

Figure 4. Syndrome Decoder for the Proposed Square (24, 16, 8) Code

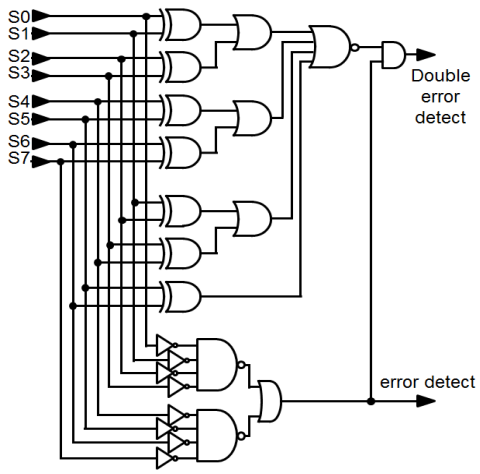

Figure 5. Error Detection Block for the Proposed Square (24, 16, 8) Code

6. Results

The results of the proposed Square code scheme are shown in Figure 6 and Table 7. Table 7 shows the overhead comparison between the Square code and conventional code. Square code scheme has the best result in overhead. In order to compare the overheads, information length is set to 64 bits. Figure 6 shows the graph of difference in parity length between the Square code and Hamming code for the same information length. This shows that the smaller the length of the information bits reduced the difference of parity length between the Square code and the Hamming code.

Table 7. Error Detection Block for the Proposed Square (24, 16, 8) Code

Code scheme	XOR stage	Data length	Area Overhead	Coverage
ATM-8 HEC	6	64	720 XOR gates	CRC
Matrix type[13]	4	64	300 XOR gates	CRC
Square code	4	64	300 XOR gates + 64 Correction block	ECC

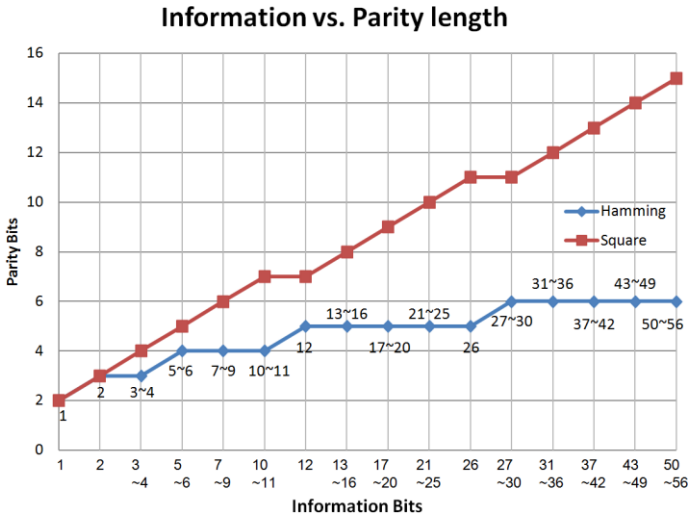

Figure 6. Error Detection Block for the Proposed Square (24, 16, 8) Code

7. Conclusion

This paper proposed a Square code which is appropriate to the WSNs. Usually, when data are transferred between the sensor nodes in a wireless sensor network, the size of the information is small because data only includes sensed information. Therefore, this paper shows the advantage of the Square code when we use it in WSNs for ECC. For the range of the small size information (<16), the parity size of the proposed Square code is almost the same compared with the Hamming code. And the proposed scheme has improved overhead than conventional codes and it has a very simple scheme in order to generate code word.

Also this paper shows the improved performance of error correcting and detecting. The Square code can detect failures that are larger than double bits error and correct failures that are larger than single bit error as shown in coverage analysis.

References

[1] J. Hill, "System Architecture Directions for Networked Sensors," Proc. Ninth Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '00), **(2000)**, pp. 93-104.
[2] M. C. Vuran, "Error Control in Wireless Sensor Networks: A Cross Layer Analysis", IEEE/ACM Transactions on networking, vol. 17, no. 4, **(2009)**, pp. 1186-1198.
[3] I. F. Akyildiz, "A Survey on Sensor Networks", IEEE Communications Magazine, **(2002)**, pp. 102-114.
[4] M. Hatler and C. Chi, "Wireless Sensor Networks: Growing Markets, Accelerating Demand", technical report, ON World, **(2005)**.
[5] S. Mukhopadhyay, "Model Based Error Correction for Wireless Sensor Networks", Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 4-7, **(2004)**.
[6] G. W. Allen, "Deploying a Wireless Sensor Network on an Active Volcano", IEEE Internet Computing, vol. 10, no. 2, **(2006)**, pp. 18–25.
[7] S. Mukhopadhyay, "Model-Based Techniques for Data Reliability in Wireless Sensor Networks," IEEE Transactions on mobile computing, vol. 8, no. 4, **(2009)**, pp. 528-543.
[8] M. Roshanzadeh, "Error Detection & Correction in Wireless Sensor Networks By Using Residue Number Systems", I. J. Computer Network and Information Security, **(2012)**, pp. 29-35.
[9] N. A. Alrajeh, "Error Correcting Codes in Wireless Sensor Networks: An Energy Perspective", Applied Mathematics & Information Sciences An International Journal Appl. Math. Inf. Sci., vol. 9, no. 2, **(2015)**, pp. 809-818.
[10] K. Koo, "A 1.2V 38nm 2.4Gb/s/pin 2Gb DDR4 SDRAM with Bank Group and x4 Half-Page Architecture", IEEE International Solid State Circuits Conference, **(2012)**, pp. 40-41.
[11] R. Logapriya, "Efficient Methods in Wireless Sensor Network for Error Detection, Correction and Recovery of Data", International Journal of Novel Research in Computer Science and Software Engineering, vol. 3, **(2016)**, pp. 47-54.
[12] O. Eriksson, "Error Control in Wireless Sensor Networks: A Process Control Perspective", ISSN: 1401-5757, UPTEC F11 030, **(2011)**, pp. 1-32.
[13] J. Lee, "Matrix type CRC and XOR/XNOR for high-speed operation in DDR4 and GDDR5", Journal of The Institute of Electronics Engineers of Korea, vol. 50, no. 8, **(2013)**, pp. 2064-2070.

Author

Joong-Ho Lee, is currently a professor of Computer Science of the University of Yongin. He received his BS degree in Electronics & Computer Engineering from University of Ulsan in 1988 and received MS degree in Electronics & Computer Engineering from University of Ulsan in 1990. He received PhD degree in Electronics & Computer Engineering from University of Ulsan in 1994. He worked as research engineer in SK-Hynix from 1994 to 2012. He was involved in DDR1, DDR2, and DDR4 SDRAM design.