

HARDWARE IMPLEMENTATION, ON A FPGA, OF GOLAY (23.12.7) ERROR-CORRECTING CODE – FOR USAGE WITHIN WSNs

Adrian-Viorel DIACONU¹, Elif Tuba ÇELIK², Ion SIMA³, Valeriu IONESCU⁴, Constantin BĂLAN⁵
^{1,3,5}LUMINA – The University of South-East Europe, IT&C Department, Bucharest, Romania
⁴University of Pitești, Faculty of Electronics, Communications and Computer Science, Argeș, Romania
^{1,2}Politehnica University of Bucharest, ETTI Department, Bucharest, Romania
adrian.diaconu@lumina.org

Keywords: data integrity, detection and error correction, Golay (23.12.7), error-correcting code, WSN, hardware and power consumption constraints, NI ELVIS II⁺, DE FPGA Board, LabVIEW

Abstract: This article presents the prototyping of a Golay (23.12.7) Error – Correcting Code encoder and decoder using a FPGA. Golay (23.12.7) CODEC's implementations were done using LabVIEW as CAD environment, having as hardware support NI ELVIS II⁺ platform equipped with DE FPGA Board. Efforts were focused on highlighting CODEC's usability within WSN by making an assessment from hardware and power consumption constraints vs. data integrity points of view. Practical test results were over-satisfactory (i.e. the circuits work very well, the balance between typical WSNs constraints and usability/necessity of the CODEC was achieved) thus proving CODEC's suitability for usage within WSNs, as a mechanism to ensure data integrity.

1. INTRODUCTION

WSNs (i.e. Wireless Sensor Networks) are composed of numerous nodes (a large number of sensor nodes and a few sink nodes) distributed over an area to collect information about the state of physical world and transmit it to interested users. If typical sensor nodes (i.e. passive ones, which can only monitor the events of interest and thus unable to react in the environment) have limited resources in terms of processing power, battery power and data storage, sink nodes have unlimited power, sufficient memory, powerful processors etc. [1], [2].

Nowadays, ensuring data security (i.e. data integrity and privacy etc.) is a core requirement in WSNs, due to deployment of a large number of sensory devices in a field. False or malicious data would result in incorrect decisions and potentially life/financial losses. One of the major security challenges for WSNs is the conflict between the limited resources (e.g. available power, computational capabilities, and storage capacity) at one hand and security requirements at the other hand [3], [4], [5] and [6].

Most of the prior works on securing sensor networks use traditional security solutions, based on cryptographic algorithms [3], [4], [6], [7], [8] which, due to their high rate of computational requirements, are too expensive (at least, in terms of power consumption) and consequently, they are not suitable for sensors [9], [10]. Few works, such as [11], [12] and [13] were focused on developing mechanisms to ensure data security on physical layer, i.e. data integrity, for modern communications systems and even fewer for WSNs.

Taking all the above into consideration, this paper presents the prototyping of a Golay (23.12.7) [14], [15], [16], [17] encoder and decoder (referred to as CODEC) as a mechanism to ensure data integrity over WSNs. In a typical WSN, denoted in Fig. 1, our CODEC is designed to work, within WSN's communication chain, between sensor nodes and sink nodes (because here the data are most likely to be altered either by malicious actions of an attacker [18], either due to environmental or intrinsic factors). Our CODEC's functional block diagram is shown in Fig. 2.

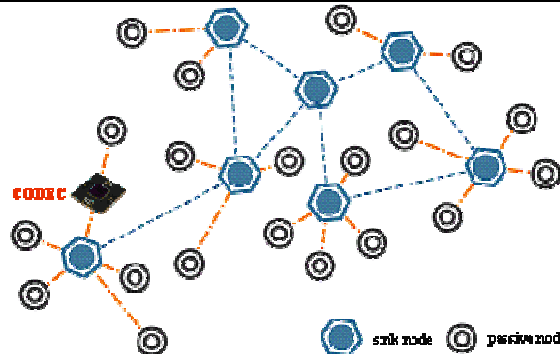


Fig. 1. Typical WSN topology and CODEC's positioning– (from functional point of view), [source image: www.myrmecore.com]

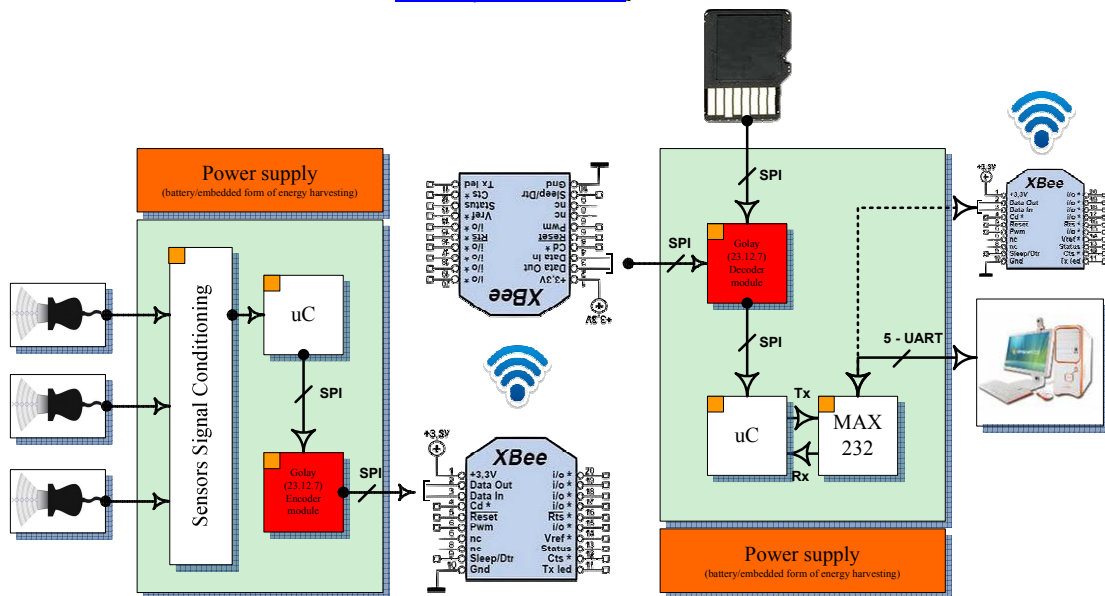


Fig. 2. CODEC's functional block diagram

As any ordinary sensor network node, ours as well, has all the typical parts: radio transceiver (i.e. XBee module, with an internal antenna or connection to an external one), electronic circuits for interfacing with the sensors (i.e. all the signal conditioning circuits, ADCs etc.), energy source (usually a battery or an embedded form of energy harvesting) and microcontroller (i.e. μC),

Beside these, in our paper point of view, supplementary circuits (i.e. the Golay encoder, for the transmitter and the Golay decoder, for the receiver) need to be added, as suggested in Fig. 2 in order to ensure data security on physical layer, i.e. data integrity.

In the block diagram presented above, the transmitter collects data from sensors (through means offered by the embedded μC , as well by sensors conditioning circuits), encapsulates them (using newly added feature - the Golay encoder) and sends them to the receiver.

The receiver can be either a BS (i.e. Base Station; end node which has high computational capabilities, unlimited available power, unlimited data storage etc.) either a sink node. On this side, the received codeword is decoded and data are processed or it is resented to a higher order sink node, within WSN.

As we will discuss later, a supplementary, available in trade, memory chip(s) or a custom designed memory chip (suggested in Fig. 2 with the aid of a microSD memory card) is needed, on the receiver side. The purpose of this memory is to store all 2048 possible correctors of Golay's error-correcting code [19]. The memory map and memory address decoding (i.e. relation between syndromes class and associated correctors) will be presented also, on 4th section of this paper, for both cases mentioned above (i.e. custom memory chip and available in trade memory chip(s)).

2. HARDWARE DESIGN

The entire work carried-out for CODEC's implementation was done using a novel hardware design methodology for digital systems, provided by LabVIEW FPGA module, and which allowed us to model the digital design, to verify it by simulation, to test and modify it very easily, thus greatly reducing development time [20], [21].

Beside the default graphical programming (i.e. dataflow source code in the form of block diagrams) a description language (i.e. Very High Speed Integrated Circuit Hardware Description Language – VHDL) was to be used, in order to implement the CODEC [22], [23].

The key starting point in hardware design stage of our CODEC, considering its functional positioning within wireless sensor's structure (i.e. it has to communicate with others embedded ICs, mainly with the μ C and transceiver), was to choose the appropriate communication protocol. Due to its presence as primary communication protocol on most μ Cs and transceivers the Serial Peripheral Interface (i.e. SPI) seemed to be the best choice, for implementation on our design. Thus, as we'll discuss later, a simplified version of this protocol, as previously described [24], was integrated on our design.

Another important challenge was to decide the most suitable dataflow structure and, after reviewing the literature [14] ÷ [17], [24] ÷ [25], the 'case' (for decisions), 'for' (for finite loops) and 'flat sequence' (for sequentiality) structures were combined, in order to achieve the best functional design of our CODEC.

Last but not least, the timing issue got the appropriate attention. While on the encoder side we had to ensure minimum computational time, although in accordance to SPI's specific timing constraints (e.g. on clock, setup and hold timing boundaries) [25] and the bit rate provided by wireless sensor's embedded μ C; on the decoder side we had to ensure a proper detection of "vital" signals such as clock (i.e. CLK) and chip select (i.e. \overline{CS}).

In the following two sub-section of this paper (i.e. 2.A. and 2.B.), the initial design of our CODEC will be presented, the final design, with slightly substantial improvements / modifications

(imposed by preliminary results - from section 3) being discussed on section 4.

A. The Golay (23.12.7) encoder module

Block diagram (i.e. dataflow source code) of the Golay encoder, including the SPI module (i.e. the bridge between the encoder and wireless transmitter), is presented in Fig. 3.

After a proper preparation of CLK and \overline{CS} signaling terminals (i.e. all to 1L, 'chip select' signal will be active on low level, respectively the 'clock' signal will be active on rising edge), the encoder waits for start signal (emulated using an external button; e.g. BTN0, in Fig. 3). When the start signal occurs, the 'flat sequence' (from inside of 'case' structure) begins to be executed. First and third sequences perform some delays, with respect to setup and hold timing boundaries, while the second one initializes the data line.

Next in line is the Golay (23.12.7) subVI, constructed also on a 'flat sequence' structure, as shown in Fig. 4. This subVI represents encoder's core.

Because polynomials multiplication is the key algorithm of Golay (23.12.7) coding, before starting subVI's explanation, reasoning on which it was built must be exhibited.

Here, a new approach is presented (instead of usual polynomial multiplication) in support of which Fig. 5 is presented.

After codeword's setup to 0x000000, each bit of the information byte is tested. If the bit is 1L, then the codeword is XOR'ed with chosen generator polynomial (1) moved to the right with N positions (where N is the position of tested bit, within the information byte) else, the XOR'ing process is skipped.

$$G(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11} \quad (1)$$

$$G(x) = 101011100011$$

At subVI's entry, the information byte to be sent (e.g. 0xAA, in our example) is converted from unsigned integer to 'Boolean Array' each of its bits being set at the input of a VHDL block. After eight steps of testing and XOR'ing (each of them being done by a different VHDL block), accordingly to algorithm described above, coded format of information byte is set at the output of the subVI.

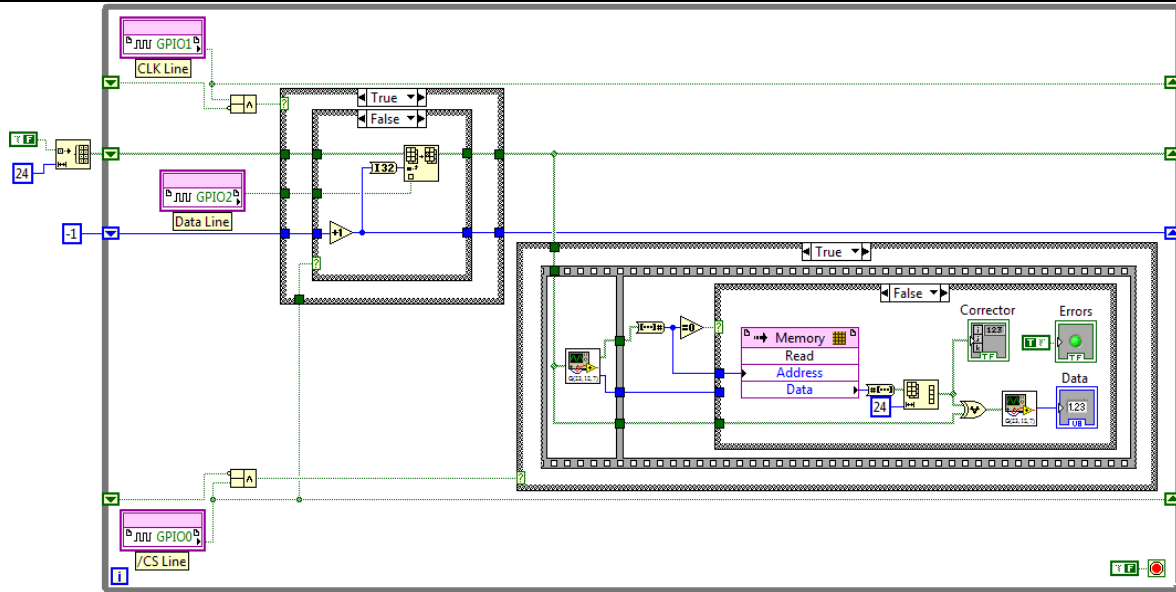


Fig. 6. LabVIEW VI - The Golay (23.12.7) decoder module (including SPI module)

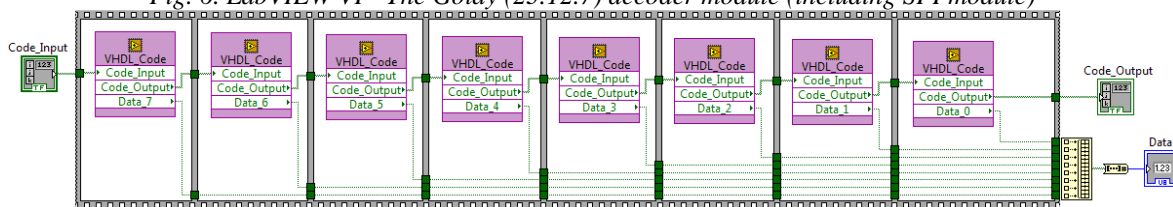


Fig. 7. LabVIEW sub VI - The core of Golay (23.12.7) decoder module

First part of main Golay (23.12.7) decoder LabVIEW VI (i.e. the left side of Fig. 6) consists in SPI receiver module (i.e. the one which takes data from wireless receiver and sends them to the decoder). SPI receiving module is constructed using two ‘case’ structures which are sensitive to external signaling lines (i.e. rising edge of CLK and low level of /CS). Each time these two conditions match data line is sampled. Samples are used to structure the codeword in an ‘Index Array’. Array’s index is incremented on each rising edge of CLK line its previous value being stored with a feedback register. This index keeps the track on number of received bits, in order to store newly received ones in the proper position. When codeword’s reception has been completed and the /CS line goes from low to high dataflow passes from SPI module to the decoding module.

It’s obvious that we had to implement few positive edge detectors into our project. Starting from basic electronic scheme from Fig. 10.a) and its input/output oscillogram from Fig. 10.b), with the aid of feedback registers and two input AND gates the LabVIEW version of our edge detectors from Fig. 10.c) was obtained.

After codeword’s reception, dataflow goes to the second ‘case’ structure which does the decoding. Here, another different point of view on classical polynomial multiplication is applied. The codeword is routed from right to left, being XOR’ed with chosen generator polynomial each time a 1L is found. If the codeword is untainted (like in first example, from Fig. 4) the result is 0x000000, and correctly decoded information (i.e. data package) is displayed; else (like in the second example, from Fig. 5) information is incorrectly decoded and the result, which this time will be different from 0x000000 (a.k.a. the syndrome), will help to find its corresponding corrector [19].

In our synthesis syndromes are expressed in binary form (i.e. at Golay subVI output), then is converted from binary string to unsigned integer. This integer gives direct access to the memory (because syndrome means address) from where the corrector is extracted. Thus, corrector is converted from integer to binary array, XOR’ed with received codeword, passed again through the Golay (23.12.7) decoder and finally correct data are displayed.

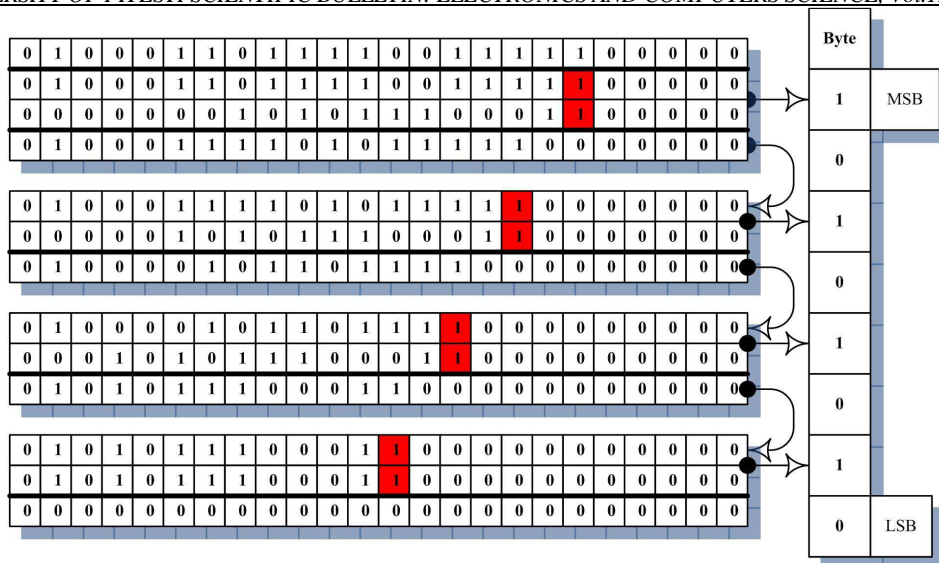


Fig. 8. Golay (23.12.7) decoding example - untainted codeword

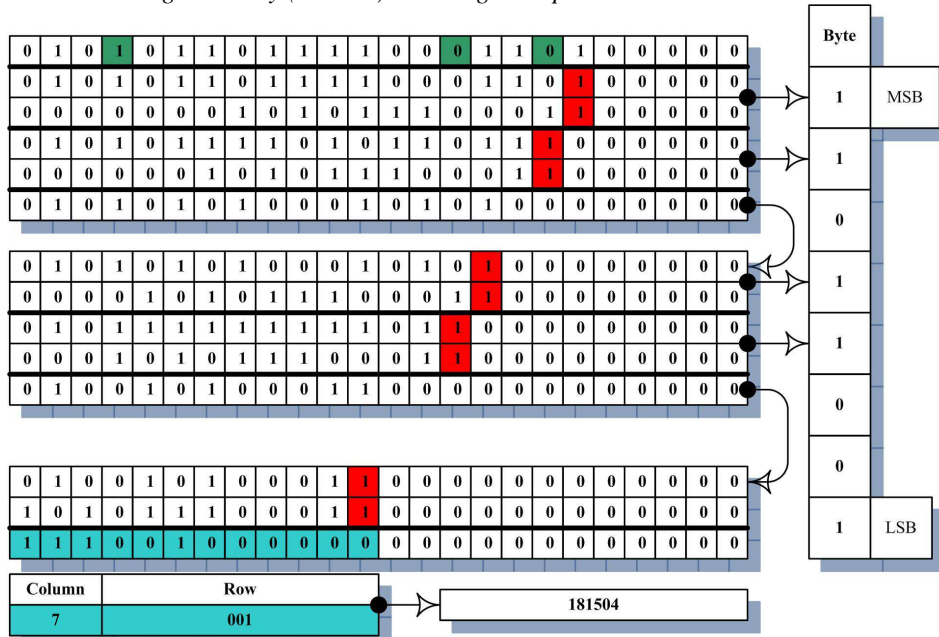


Fig. 9. Golay (23.12.7) decoding example - tainted codeword

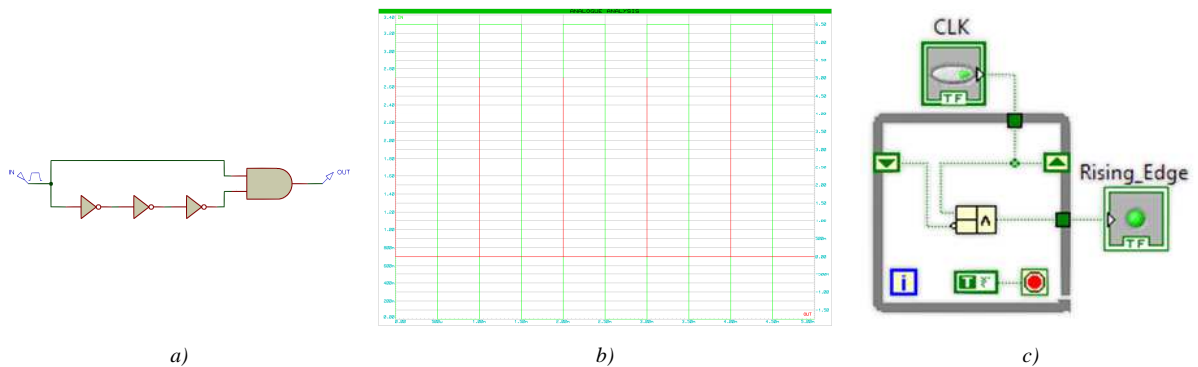


Fig. 10. Edge detector – electronic schematic a), input and output signals b) and LabVIEW implementation c)

3. PRELIMINARY RESULTS

After hardware synthesis of our CODEC, the following reports were generated:

TABLE 1: HARDWARE SYNTHESIS REPORTS

Golay (23.12.7) encoder	
Report item	Performance
Min. clock period (Max. clock frequency)	8.231 ns (121.758MHz)
Max. path delay from/to any node	2.498 ns
Max. net delay	0.713 ns
Total number of slices	461/4656
Total number of FFs	592/9312
Total number of LUTs	614/9312
Golay (23.12.7) decoder	
Report item	Performance
Min. clock period (Max. clock frequency)	9.896 ns (101.051MHz)
Max. path delay from/to any node	3.055 ns
Max. net delay	0.953 ns
Total number of slices	605/4656
Total number of FFs	800/9312
Total number of LUTs	763/9312

As can be seen from the two reports, the chip area occupied in a Xilinx Spartan 3E FPGA is very small (i.e. both encoder and decoder uses less than 10% of the FPGA's resources). This performance conjures integration of our system on the same chip with other modules.

The hardware synthesis was followed by measurements on the computational speed. These measurements have yet revealed another major advantage namely – the decrease of computing time over a software solution and also over other algorithms design to protect data at the physical layer. Comparison software implementation was done on an „Intel CPU T3200/2GHz” platform, while reference algorithms were fully compared on [10]. Results presented on Table 2 reveals that our solution's hardware synthesis is at least with one to three orders of magnitude faster than referred encryption algorithms and few times faster than the software implementation.

TABLE 2: COMPUTATIONAL SPEED COMPARISONS

	Processing time for	
	Coding	Decoding
Golay (23.12.7) CODEC	273.15ns	412.83ns
Software implementation	685.55ns	1.215µs
Other solutions [10]	5.890µs ÷ 345.886µs	

Fig. 11 through Fig. 13 are showing some qualitative and quantitative measurements over a

codeword received by our decoder. From these figures we can conclude that with an average 8µs/codeword, our CODEC is able to compute approximately 120k codeword/s.

This amount of data (in terms of codewords) that can be computed and sent out by our CODEC satisfies requirement of any general WSN.

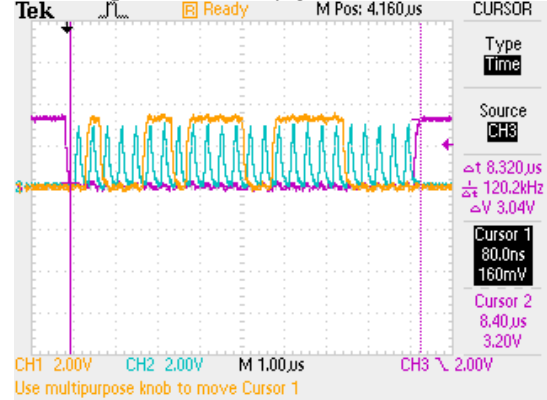


Fig. 11. Image showing the output of Golay (23.12.7) encoder: \overline{CS} (on CH3), CLK (on CH2) and data (on CH1) signals

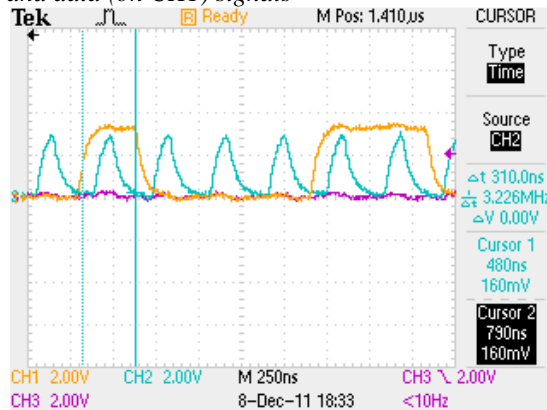


Fig. 12. Image showing measurement of Data bit-rate/duration

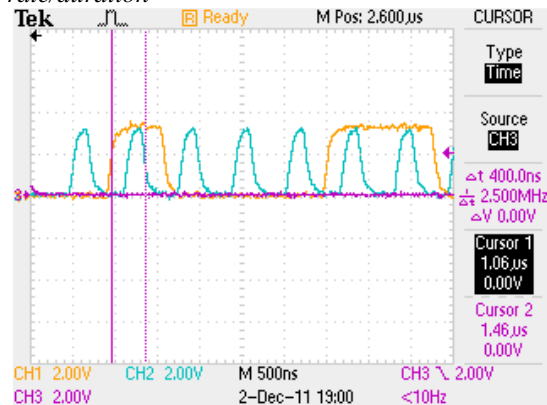


Fig. 13. Image showing measurement of CLK's frequency

4. "FINAL TOUCHES"

In this section of the paper we'll propose few solutions for some important issues that have risen in design and exploratory phases of our project.

The two most important issues, which worth our attention (in order to give to our CODEC suitability for usage within WSNs), are related to data storage requirements and power consumption.

A. Power consumption issue

It's obvious (both from literature and the above figures) that the number of bits to be sent over the communication chain is tripled by our CODEC, and because more bits to be sent means more power consumption, countermeasures need to be found.

It seems that most suitable, for our field of application, is the solution which consists in a physical layer signal modulation scheme that includes M-ary modulation (i.e. sending multiple bits per symbol) [26], where M should be at least equal with 2. Our chosen XBee transceiver offers OQPSK (i.e. sometimes, so called SQPSK) thus, power consumption issue is solved.

Another interesting approach, on solving power consumption issue, taking into account the amount of energy required by our CODEC, is to use energy harvesting power sources [27], [28]. Such power sources would meet the energy requirements of our CODEC, for a period of time roughly the same as wireless sensor's life time.

B. Data storage requirements issue

In terms of data storage, the most suitable approach is the usage of external storage devices.

Beside the fact that memory's synthesis into available resources of the FPGA isn't such a good idea (from technical and practical points of view), the amount of memory required exceeds remaining resources. Therefore, other solutions have been studied, two of them being discussed in the following.

First idea that pops-up in mind is to design a memory chip according to the project needs (i.e. 2k·24bit ROM). So, as the FPGA can work with words of 24-bit in length, each corrector can be read entirely, at once (i.e. based on a single address) immediately after syndrome calculation. This solution has its weaknesses, namely the

necessity to integrate another module to ensure proper communication between Golay decoder and external memory.

Regardless of strengths and weaknesses, taking into consideration the fact that another SPI module can be easily fitted into our decoder design, the memory map (as a relation between the syndromes class and associated correctors) is presented Fig. 14.

Another approach on this issue is to use an available, in trade, memory chip (e.g. 24C08 or 24LC08). This solution has its weaknesses also, requiring integration of an I²C communication module, along with supplementary circuitry for an appropriate addressing.

For these memory chips, memory map will be totally different, taking into account that they have the word in length of 8 bits. Thus, between two consecutive syndromes (i.e. addresses) will be an offset, as depicted in Fig. 15.

As we saw in hardware design section, for testing purposes, the memory was synthesized into FPGA (with only few correctors, e.g. from rows 000÷005 [19]). In practical tests having 24LC08 type memory chips at our disposal we had to eliminate this internal memory. Thus, the project has undergone small modifications. An I²C communication module was synthesized along with the Golay (23.12.7) decoder, in order to access the external memory. Beside memory map of 24LC08 presented in Fig. 14, the manner in which the custom designed ROM should be programmed is presented in Fig. 15.

5. CONCLUSIONS

In this paper we managed to present the prototyping of a Golay (23.12.7) error-correcting CODEC, using one of the most novel design methodologies for digital systems. Implemented in targeted Xilinx Spartan 3E FPGA, the circuit works very well, any 3 bit error in any position of 24 bit codeword being corrected.

Designed with respect towards WSN's specific constraints our CODEC proves itself to be suitable for usage within any WSN, as a mechanism to ensure data integrity.

Future research works are directed towards investigation BER-SNR performances of WSNs, in which all sensor nodes are equipped with our error-correcting CODEC.

6. REFERENCES

- [1]. Culler, D., Estrin, D., Srivastava, M., "Overview of sensor networks", IEEE Computer Magazine, Volume No. 37, Issue No. 8, January 2004, pp. 41÷49.
- [2]. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E., "Wireless sensor networks: A survey", IEEE Communications Magazine, Volume No. 40, Issue No. 8, August 2002, pp. 102÷114.
- [3]. Pathan, A.S.K., Hyung, L., Hong, C.S., "Security in Wireless Sensor Networks: Issues and Challenges", Proceedings of 8th IEEE International Conference on Advanced Communication Technology, Phoenix Park, Korea, February 2006, Volume No. 2, pp. 1043÷1048.
- [4]. Vaseashta, A., Vaseashta, S., "A Survey of Sensor Network Security", Sensors & Transducers Journal, Volume No. 94, Issue No. 7, July 2008, pp. 91÷102.
- [5]. Kamel, I., Juma, H., "A Lightweight Data Integrity Scheme for Sensor Networks", Sensors – Open Access Journal, Volume No. 11, Issue No. 4, April 2011, pp. 4118÷4136.
- [6]. Muazzam, A.K., Ghalib, A.S., Muhammad, S., "Challenges for Security in Wireless sensor Networks (WSNs)", World Academy of Science, Engineering and Technology, Issue No. 80, August 2011, pp. 390÷396.
- [7]. Perring, A., Przydatek, B., Song, D., "SIA: Secure Information Aggregation in Sensor Networks", Journal of Computer Security, Volume No. 15, Number 1, Special Issue on Security of Ad-hoc and Sensor Networks, 2007, pp. 69÷102.
- [8]. Ren, X., Yu, H., "Security Mechanisms for Wireless Sensor Networks", International Journal of Computer Science and Network Security, Volume No. 6, Issue No. 3, 2006, pp. 151÷161.
- [9]. Iyengar, S.S., Durresi, A., Paruchuri, V., Kannan, R., "Data Integrity Protocol", International Journal of Distributed Sensor Networks, Volume No. 1, Issue No. 2, 2005, pp. 205÷214.
- [10]. Venugopalan, R., Ganesan, P., Peddabachagari, P., Dean, A., Mueller, F., Sichitiu, M., "Encryption Overhead in Embedded Systems and Sensor Network Nodes: Modeling and Analysis", Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems, San Jose, CA, USA, October 2003, pp. 188÷197.
- [11]. Kusumawardani, S.S., Sutopo, B., "Designing 1 bit Error Correcting Circuit on FPGA Using BCH Codes", Proceedings of International Conference on Electrical, Communication, and Information, CECI 2001, Jakarta, 7÷8 March, 2001.
- [12]. Ionescu, L.M., Anton, C., Tutănescu, I., Mazăre, A., Șerban, G., "Hardware Implementation of BCH Error-Correcting Codes on FPGA", International Journal of Intelligent Computing Research, IJICR, Volume No. 1, Issue No. 3, June 2010, pp. 148÷153.
- [13]. Anton, C., Ionescu, L.M., Tutănescu, I., Mazăre, A., Șerban, Gh., "FPGA – implemented CRC Algorithm", International Conference "Applied Electronics", University of West Bohemia, Pilsen, Czech Republic, 9÷10 September, 2009, pp. 25÷30.
- [14]. Hoffman, D.G., Leonard, D.A., et.al, "Coding Theory: the Essentials", Marcel Dekker, Inc., 1991.
- [15]. Rhee, M.Y., "Error-Correcting Coding Theory", McGraw-Hill, 1989.
- [16]. Lin, S., Costello, D.J., Jr., "Error Control Coding: Fundamentals and Applications", Prentice Hall, Inc., 1983.
- [17]. Peterson, W.W., "Error-Correcting Codes", MIT Press, 1961.
- [18]. Uppuluri, P., Basu, S., "LASE: Layered approach for sensor security and efficiency", Proceedings of the 2004 IEEE International Conference on Parallel Processing Workshops, Montreal, Quebec, Canada, 15÷18 August, 2004, pp. 346÷353.
- [19]. Baltă, H., "Referat nr. 1. – Coduri utilizate în sistemele de transmisiuni cu spectru împărțiat", pp. 93÷96, unpublished.
- [20]. Singh, S., Rattan, S.K., "Implementation of a Fuzzy Logic Controller on an FPGA using VHDL", 22nd International Conference of the North American Fuzzy Information Processing Society, Chicago, Illinois USA, 24÷26 July, 2003, pp. 110÷115.
- [21]. Mahora, S., Kehtarnavaz, N., "Digital Signal Processing Laboratory: LabVIEW-Based FPGA Implementation", BrownWalker Press, Boca Raton, Florida, USA, 2010.
- [22]. XIOS Hogeschool Limburg, "Embedding VHDL code in a Xilinx Spartan 3E V1", accessed on [01.05.2012]: <http://pwo.fpga.be/LabVIEW>.
- [23]. Pedroni, V.A., "Circuit Design with VHDL", MIT Press, 2004.
- [24]. www.ni.com, "Implementing SPI Communication Protocol in LabVIEW FPGA", published date: 18th of March, 2010, accessed on [10.05.2012].
- [25]. www.ni.com, "Understanding the SPI Bus with NI LabVIEW", published date: 20th of December, 2011, accessed on [10.05.2012].
- [26]. Shih, E., Cho, S., Ickes, N., Min, R., Sinha, A., Wang, A., Chandrakasan, A., "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks", Proceeding of the 7th Annual International Conference on Mobile Computing and Networking, Rome, Italy, 16÷21 July, 2001, pp. 272÷287.
- [27]. Winston, K.G. Seah, Eu, Z.A., Tan, H.P., "Wireless Sensor Networks Powered by Ambient Energy Harvesting (WSN-HEAP) – Survey and Challenges" Proceedings of the 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronics Systems Technology, Wireless VITAE, Aalborg, Denmark, 17÷20 May, 2009, pp. 1÷5.
- [28]. Chalasani, S., Conrad, J.M., "A Survey on Energy Harvesting Sources for Embedded Systems", IEEE Southeastcon, 2008, pp. 442÷447.

