HARDWARE IMPLEMENTATION OF A SINGLE BIT ERROR CODE CORRECTION

Constantin ANTON, Gabriel IANA, Gheorghe ŞERBAN
Ion TUTANESCU, Petre ANGHELESCU
University of Pitesti
Str. Targul din Vale, nr. 1, Pitesti, Romania
constantin.anton@upit.ro, gabi@upit.ro, serban.gheorghe@upit.ro
ion.tutanescu@upit.ro, petre.anghelescu@upit.ro

Keywords: Hamming code, hardware structure, communication, error correction, FPGA implementation.

Abstract. *In this paper is hamming code proposed implementing a structure for reconfigurable hardware for error correction bits on a line of communication. Algorithms for implementing the hamming code is made on a structure as simple and is aimed at the trials of code/decode the information to perform at a speed as much as possible, without the special hardware consumes resources. They are made functional simulations of implemented module and comparative results speed/resources occupied for various lengths of sequences.*

INTRODUCTION

In communications, transmission systems over must receive packets of information data without error, at high speeds and an architecture with a small complexity with a low costs. In cases where the environment of transmission is noisy, correcting for packages of data can use a method for retransmission of those packages which are incorrect. This method of interrogation and the retransmission data involve stroke large stakes in the transmission of data, in many cases being inefficient. To avoid this disadvantage, in some cases is used hamming code of errors correction.

Hamming Codes are generally used in computing, telecommunication, and other applications including data compression, and turbo codes. Also, the Hamming codes are used, for example, as forward correcting cedes in the Bluetooth standard, and to protect data stored in semiconductor memories. Hamming codes are used very well for low-cost, low-power application, and when decoded iteratively, they can approach the capacity of an additive white Gaussian noise (AWGN) channel.

The algorithm for calculating hamming code is one attractive, low and the complexity can be easily implemented in digital structures. For these reasons are fairly attractive with the implementation of some types of applications of high speed, the communications field.

1. HAMMING CODE THEORY

Hamming codes permits correcting single-bit errors. At k bit of information bits its used m the number of check. Because the m check bits must check themselves as well as the information bits, the value of p (parity check bit vector), interpreted as an integer, must range from 0 to $m+k$ which is $m+k+1$ distinct values. All, the m bits can be distinguish at 2^m cases, as shown in equation 1.

$$2^m \geq m+k+1 \qquad (1)$$

It applies to any single error correcting (SEC) binary Forward Error Correction (FEC) bloc codes in which all of the transmitted bits must be checked.

The rate of the code defined the relationship k/n involving processing a higher number of bits of information to obtain rates

close to 1. The hamming code has the power of correction little fact that involves the use of number of bits informational small. According to the noise of channel of communication, (p which means error probability of a symbol), we choose number of informational bits for noisiest channels ($p <<$), we chose to grow k . In figure 1, it is presented an example where $p = 0.01$ (noisy channel), k can't be bigger as 11bits.

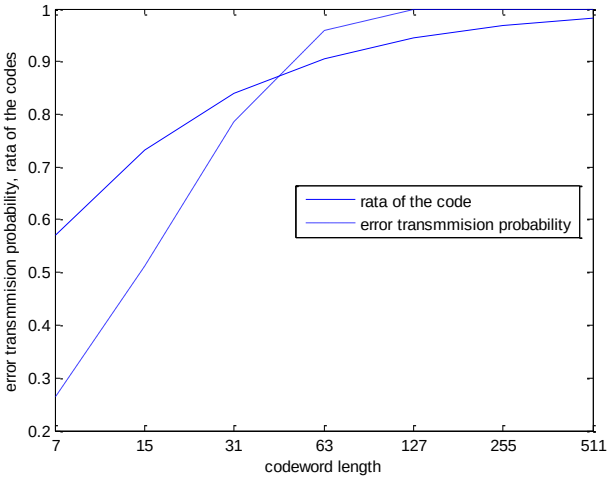

Fig. 1. The calculation errors and probability rate code word depending on length of the Code

The check bits will be interspersed among the information bits in a manner described below.

Putting the check bits in positions 2^m has the advantage that they are independent. That is, sender can compute p_0 independently of p_1, p_2, … and more generally, it can compute each check bit independently of the others.

As an example, let us develop a single error correcting code for $k = 4$, solving (1) for m gives $m = 3$, with equality holding. This means that all 2^m possible values of the check bits are used, so it is particularly efficient. A code with this property is called a perfect code.

This codes is called the (7,4) Hamming code, which signifies that the code length is 7 and the number of information bits is 4.

In the figure 2 is plotted the throughput channel function probability of error where shown the efficiency of Hamming code.

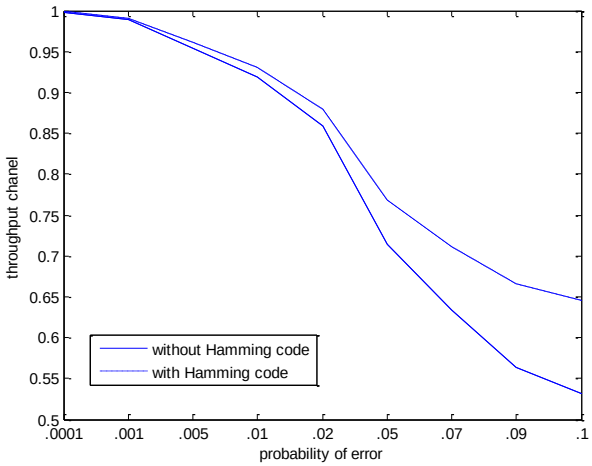

Fig. 2. Throughput channel function probability of error - Hamming (7,4).

The positions of the check bits p_i and the information bits p_i are shown in figure 3.

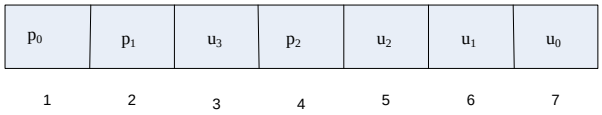

Fig. 3. The positions of the check bits in the information bits string.

2. THE RECONFIGURABLE HARDWARE

2.1. Overview

The reconfigurable hardware comes from idea to change the internal interconnection of digital devices. First was introduced the Programmable Logic Devices (PLDs), Programmable logic array (PAL), Generic Array Logic (GAL) device that is a PAL devices but could be erased and reprogrammed. The needed of finite state machine implementation, was introduced the CPLD (Complex PLD) devices. CPLDs have a complexity that can replace very much of logic gates but these devices are not enough to implement complexes algorithm. Then was introduced Field Programmable Gate Array (FPGA). These devices have a different architecture of reconfiguration in comparison with CPLD. The FPGA is built using a grid of configurable logic blocs interconnected by array of wire. Those devices are capable to implement very complex algorithms.

The future tendency is it the developing Configurable System on Chip (CSoC). In this devices can be immersed the cores of processors and is combined the sped of FPGA data flow processing and capability of multiple task managing with the core processors

The implementation on hardware algorithms is made trough hardware description as Verilog, and especially, VHDL. Recently, FPGAs have been used for reconfigurable computing when the main goal is to obtain high performance at a reasonable cost out of hardware implemented algorithms.

The main advantage of FPGAs is their reconfigurability, i.e., they can be used for different purposes at different stages of a computation and they can be, at least partially, reprogrammed on run-time.

2.2. The FPGA architecture

The architecture of Xilinx FPGA families consists of five fundamental functional elements:

- Configurable Logic Block (CLB) and Slice architecture;
- Input/Output Blocks (lOBs);
- Block RAM;
- Dedicated Multipliers and;
- Digital Clock Managers (DCMs).

The configurable Logic Blocs (CLBs) are the most important and abundant hardware resource of an FPGA. They are typically utilized for both, combinatorial and synchronous logic design. Each CLB is composed of four slices (figure 4).

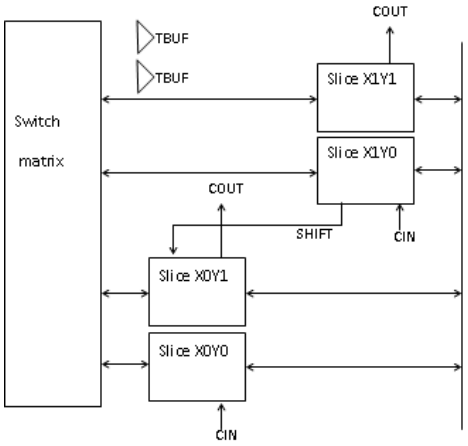

Fig. 4. Structure of Configurable logic blocs.

The slices are grouped by pairs and each pair is organized by a column with independent carry chain. All four slices have the following common elements: two Look-Up Tables (LUTs), two type D flip-flops, multiplexers, logic circuits for carry handling and arithmetic logic gates. Both, the left and right pair of slices utilize those elements for providing logic functions, arithmetic and ROM. Besides that, the left pair supports two additional functions: data storage using a distributed RAM and 16-bit shift register functionality.

3. THE IMPLEMENTATION OF HAMMING CODE

The hamming code, on the reconfigurable structures, can be implemented through three possibilities. First is a description of type data

flow, the second implementation is with finite state machine (FSM) and third by a full behavioral description. Using first type of description, hamming code module would be implemented full concurrent and it makes sense only if the dates of entry are supplied in parallel form. This description would involve only combinational subcomponents.

If the data are taken in series strings, can be achieved sequential synchronous circuits. In this case is use second and third method. By finite state machine description, is one of the most used to achieve implementation of hamming code. This has the facility to interpret step by step hamming code. However it has the disadvantage that occupies much area of a FPGA structure which is not convenient when is needed the implementation on a low cost simple FPGA structure.

The black box of module which was implemented is presented in following figure:

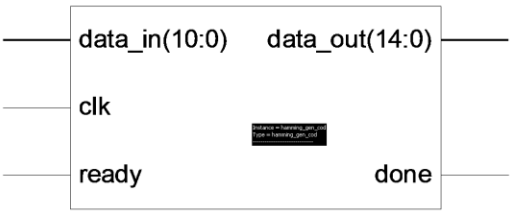

Fig. 5. Black box of hamming code module.

The synchronization of hamming module is carried after the *clk* signal. The signals *ready* and *done* are used for handshaking with another. The input ports *data_in*, i.e output data *data_out* are a single bit, the circuit being placed in a serial communication.

For example, at (15,5) Hamming code, when appear a rising edge on *ready* port, the module begin counter with third bit because the first and the second are the check parity bits. After 15 bits, serial transmitted, the circuit gives a strobe on *data_out* port. Diagram signals are as follows:

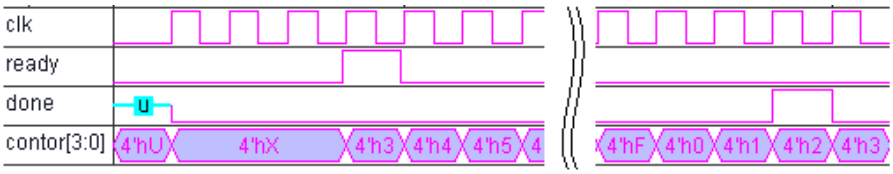

Fig. 6. Signal diagram of Hamming module.

Internally, the Hamming module contains 30 cells of memory that represents 30 bits. Practically may was two string of every 15 bits. He used these two strings in parallel, one for loading data and check parity bits computing and the second for data transmission check parity bits (figure 7).

The decode Hamming module is use the same principle. Physically, when received data series, is calculated on the parallel ones 5 bits par. When it sent the last bit, the check parity bits are placed in a string on appropriate positions.

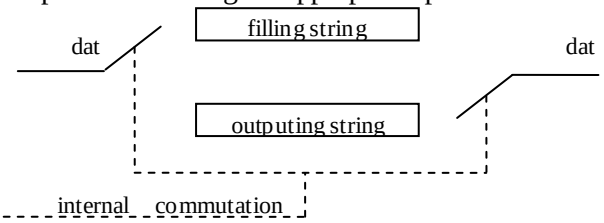

Fig. 7. Commutation of internal strings.

A date string format is transmitted but at the same time the string which had been sent now is filled with a new string of serial data entry.

Hamming algorithm code has been described in VHDL language. The source code is organized on 5 modules. They are used, for synchronization of input signals, check parity bits, switching strings and sending data.

4. PERFORMANCE AND EXPERIMENTAL RESULTS

Haming code module was implemented on a structure FPGA, type SPARTAN 3. Its performance for a (15.5) hamming code is:

```
Minimum period: 5.778ns (Max. Freq.:
173.070MHz)
Minimum input arrival time before clock:
6.099ns
Maximum output required time after clock:
7.165ns
```

The area consumed of FPGA is presented next:

```
Number of Slices:          46 out of 3584     1%
Number of Slice Flip Flops:58 out of 7168     0%
Number of 4 input LUTs:    51 out of 7168     0%
Number of GCLKs:            1 out of    8    12%
```

5. CONCLUSIONS

Hamming codes are used on channels as little as it would be with noisy as memories, lines of high-speed communications. These codes were implementing hardware relatively simple allowing communications systems FEC high-speed but with a correction is their relatively small.

Even if they were discovered newer other methods, they still remain actually and hardware implementation is best suited for least noisy high-speed channels of communications.

REFERENCES

[1]. Specification of the Bluetooth System, "Core system package," Vol. 2,Nov. 4, 2004

[2]. K. Thaller and A. Steininger, "A transparent online memory test for simultaneous detection of functional faults and soft errors in memories, "*IEEE Trans. on Reliab.*, vol. 52, no. 4, pp. 413–422, Dec. 2003

[3]. C. Desset and A. Fort, "Selection of channel coding for low-power wireless systems," in *Proc. VTC*, 2003, vol. 3, pp. 1920–1924

[4]. Hamming W.Richard. Coding and Information Theory Prentice-hall, chapter 3, 1983

[5]. Patrick G. Farrell, Jorge Castineira Moreira. Essentials of error-control coding, John Wiley &Sons, Ltd 2006, chapter 2

V. K. Wei, "Generalized Hamming weights for linear codes," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1412–1418, Sept. 1991