Automatic Implementation of Low-Complexity QC-LDPC Encoders

Georgios Tzimpragos†‡, Christoforos Kachris†, Dimitrios Soudris* and Ioannis Tomkos‡

*School of Electrical and Computer Engineering, National Technical University of Athens, 15780 Zographou Campus, Athens, Greece
Email: {getzim, dsoudris}@microlab.ntua.gr
‡Athens Information Technology, 19002 Peania, Athens, Greece
Email: {kachris, itom}@ait.edu.gr

Abstract—Low Density Parity Check (LDPC) codes are a special class of error correction codes widely used in communication and disk storage systems, due to their Shannon limit approaching performance and their favorable structure. In this paper an Electronic Design Automation tool for the generation of synthesizable VHDL codes, implementing low-complexity Quasi-Cyclic LDPC (QC-LDPC) encoders is presented. The designs generated by the developed tool have been proved to exhibit hardware savings and greater throughput as compared to other published QC-LDPC encoder implementations and are based on a design methodology, where the signals in many cases are hard-wired in the LUTs and the cyclic-shifters and block-memories conventionally used, are eliminated. The presented tool also offers the advantage of providing designers with the ability to study the trade-offs in maximum clock frequency, throughput, resources utilization and power consumption, between architectures with different design parameters, enabling rapid Design Space Exploration.

I. INTRODUCTION

Error control coding (ECC) is a discipline of Information Theory, introduced by Claude Elwood Shannon in 1948 [1]. In his landmark paper Shannon showed that channel noise limits the transmission rate, not the error probability. Hence, it is possible to design an error-free communication system using error control coding. One of the most widely used error control methods is Forward Error Correction (FEC). The main idea besides FEC is to provide the transmitter with ways to encode data signals in a redundant way following certain relations, in order to enable automatic error detection and correction in the received signals.

Low Density Parity Check codes are a special class of forward error correction codes widely used in communication and disk storage systems, due to their Shannon limit approaching performance and their favorable structure. Compared with their counterparts, LDPCs in many cases demonstrate better characteristics, such as parallelism in decoding and simple computation operations while having high performance. Although, such codes have been used as a coding scheme for a while, the development of FEC solutions suitable for systems with strict speed, area and power requirements remains challenging.

In this paper, we focus on the development of an Electronic Design Automation (EDA) tool for the generation of synthesizable VHDL codes, implementing efficiently Quasi-Cyclic LDPC (QC-LDPC) encoders. The generated codes are based on a hard-wire oriented methodology, where the use of costly block memories and multipliers/shift registers is eliminated, enabling high throughput and low resources consumption. In literature, to the best knowledge of the authors of this paper, the presentation of such an EDA tool has never been given.

Overall, the main contributions of this paper are the following:

- A hard-wire oriented architecture for QC-LDPC encoding,
- An EDA tool for the automatic generation of such low complexity and high performance designs,
- A performance evaluation of the generated codes in terms of maximum clock frequency, throughput, area and power consumption.

The remainder of the paper is organized as follow. In Section II, after a brief introduction to QC-LDPC codes, an efficient algorithm for their encoding is reviewed. A method for optimized hardware multiplication by constant matrices in (Galois field) GF(2) is described in Section III and a generalized architecture of QC-LDPC encoders is introduced, as well. The developed EDA tool/Automatic VHDL Generator is also presented in Section III. In Section IV, implementation results of the generated designs are provided, demonstrating the advantages offered by the presented tool for rapid Design Space Exploration. Finally, concluding remarks are given in Section V.

II. ENCODING ALGORITHM FOR QC-LDPC CODES

Binary QC-LDPC codes are a special subclass of LDPC codes. Their parity check matrix can be represented as an array of square $z \times z$ submatrices over GF(2), where each submatrix is either a zero or a permutation matrix. These codes have recently received much attention and are considered as a promising candidate coding scheme for many systems, due to their advantageous structure and low memory requirements, as compared with conventional LDPCs. Some communication standards, such as IEEE 802.11 and IEEE 802.16, have also adopted QC-LDPCs as error correction codes for their channel
coding scheme and support various code rates and block lengths. The codes included in the mentioned IEEE standards own the special property of the parity check matrices of Block-type LDPCs (B-LDPCs) and their encoding can be done in an efficient way, without reference to the code generation matrix, as described in [2]. According to this paper, the encoding algorithm of B-LDPCs can be summarized in the following steps:

- Step 1: Compute $A s^T$ and $C s^T$
- Step 2: Compute $ET^{-1} As^T$
- Step 3: Compute $p^T = ET^{-1} As^T + Cs^T$
- Step 4: Compute $p^2T$ by $T p^2T = As^T + B p^T$

where $A, B, C, D, E$ and $T$ are submatrices of the parity check matrix $H$, with size $(M_b - 1)z \times K_b$, $(M_b - 1)z \times z$, $z \times K_b$, $z \times z$, $z \times (M_b - 1)z$ and $(M_b - 1)z \times (M_b - 1)z$, respectively. $M_b$ and $N_b$ denote the number of rows and columns of the base matrix of $H$ and $K_b = N_b - M_b$.

III. IMPLEMENTATION

A. Architecture

In this subsection we present a generalized architecture for QC-LDPC encoding, implementing the algorithm described above. The proposed design is based on the novel hard-wire oriented methodology introduced in [3], leading to efficient QC-LDPC encoders with high throughput and low resources consumption implementations. In general, these methods focus on the optimization of hardware multiplication by constant binary matrices. The following example should make things more clear.

Assuming that we want to perform the matrix multiplication:

$$ q = L x^T $$

(1)

where $L$ is a $z \times 3z$ matrix consisting of $z \times z$ $P^{a_1}$ submatrices, $x$ is a $3z \times 1$ matrix and $q$ is a $z \times 1$ matrix.

$$
\begin{bmatrix}
q_{z-1} \\
q_{z-2} \\
\vdots \\
q_1 \\
q_0
\end{bmatrix} =
\begin{bmatrix}
P^{a_2} & P^{a_1} & P^{a_0}
\end{bmatrix}
\begin{bmatrix}
x_{3z-1} \\
x_{3z-2} \\
\vdots \\
x_1 \\
x_0
\end{bmatrix}
$$

(2)

According to conventional approaches, this operation is implemented with the use of $z$-bit cyclic shifters (for the matrix multiplication), as the $P^{a_i}$ matrices are circulant. However, in the proposed solution the matrix $L$ is regarded fixed and thus we can skip the shift operation and connect the signals to the right LUTs (modulo-2 addition) directly as depicted in Figure 1. In that way, the number of required resources is reduced and additionally the block-memories conventionally used, are eliminated. Therefore, it is obvious that the presented methodology leads to more compact designs with much lower hardware complexity.

Since the parity check matrix of QC-LDPC codes is composed of circulant shifted identity and zero matrices, the key problem of designing the encoder is the efficient multiplication of the input data by a number of circular shift unit matrices. Hence, if the parity check matrix is defined as constant, we can proceed with an implementation based on the described methodology, approaching the efficiency of hard-wired solutions. To enable the support of different code rates and block lengths within this framework, use of the reconfigurable nature of FPGAs should be made. Figure 2 illustrates an architecture overview of the Block-type LDPC encoder, according to this design methodology.

As it has been mentioned above, the arithmetic operations of the binary LDPC codes are in GF(2) (modulo-2 addition). Therefore, all the generated blocks in the depicted architecture are used as XOR operators. However, the number of LUTs and registers required for the implementation of each of them is not fixed as the number of their inputs varies.

B. EDA tool

The main contribution of this work is the development of a software tool for implementing automatically low-complexity QC-LDPC encoders onto FPGAs. The description language used as an entry tool to model the hardware architecture is the VHDL. The developed tool requires as inputs the size of the
subblocks $z$ and the LDPC matrix and exports the associated RTL code.

Following the design methodology described earlier, the presented tool finds at first the number of required blocks to carry out the arithmetic operations (XOR operators in our case) at each step of the encoding algorithm. Then it searches for the minimum number of required LUTs to implement these blocks, and connects them properly (Figure 1). For example, to carry out the $Cs^T$ function of Step 1 (Figure 2) for the parity check matrix given by the IEEE 802.11 standard, with code rate $R = 5/6$ ($M_b = 4$ and $N_b = 24$), $z$ XOR operators are required, as $C$ is a $z \times K_b$ matrix.

Due to the fact that these XOR operators have $K_b = N_b - M_b = 20$ bits as input, 4 LUT components (assuming that the target reconfigurable medium has 6-input LUTs) should be generated for the implementation of each (Figure 4). Next, as denoted in Figure 3, a commercial software tool, i.e. the Xilinx ISE, is used for the design simulation, synthesis, mapping, and place and route onto FPGA devices. With the use of this tool, the user gets reports on the implementation results and the generated bitstream file, acquired for programming the FPGA device.

**IV. IMPLEMENTATION RESULTS AND COMPARISON**

This section is devoted to the presentation of IEEE 802.11 LDPC encoder’s implementation results. A comparison in terms of throughput and area efficiency between existing solutions and the proposed one ("hard-wired") is provided in [3], proving the latter’s superior performance. A performance evaluation of the generated codes in terms of throughput, area and power consumption follows. All designs (12 LDPC matrices are provided by the IEEE 802.11 standard) included in the following comparisons are generated by the described tool. The target reconfigurable medium is a Xilinx Virtex5 (xc5vlx20t).

Figure 5 provides information about how any change in the code rate or and the subblock size affects the maximum clock frequency of the generated design. As it was expected, the design’s maximum clock frequency diminishes as the subblock size increases for the same code rate (bigger subblock sizes lead to bigger designs). Additionally, it is noticed that greater operating frequencies are achieved for the smaller rates.

Figure 6 shows the relation between the number of required LUTs per input bit and the subblock size and the code rate. It is obvious that the value $z$ of the subblock size does not affect this metric at all, whereas it depends on the code rate. More specifically, the number of required LUTs per input bit is changing approximately linearly with the code rate. Therefore, we can come to the conclusion that the maximum number of the required resources is predictable and an accurate area estimation could be made even before the code generation.

Figure 7 depicts the dynamic power consumption of each design depending on the rate and the subblock size. To measure the power consumption, each of the designs has been
simulated using the same data as input, while operating at its maximum clock frequency. The switching activity of the logic elements has been extracted from the simulation tool and then used along with the associated netlist to feed the Xilinx Power Estimator [4] and calculate the total power consumption.

![Fig. 7. Dynamic power consumption per input bit.](image)

The dynamic power consumption has been normalized to power per input bit. As can be seen from the above figure, the lowest power per bit is achieved when the rate is 0.75. In that case, the power consumption is almost the same for any subblock size. When the rate is 0.5 or 0.66 more redundant bits are added to the original message as compared to rate 0.75 and thus better error correction is achieved, but that happens at the expense of additional power consumption per bit. The fact that the generated designs for code rate 0.83 dissipate more power per bit than those of rate 0.75 is noticeable, as it differs from what we have observed in the previous charts, where the value of the associated metrics is decreasing as the code rate increases. Considering that for code rate 0.83 fewer LUTs are required, we come to the conclusion that the main cause of this anomaly should be wiring.

![Fig. 8. Implementation results exploration, when throughput and the total number of LUTs of the mapped designs are defined as criteria.](image)

Furthermore, as can be seen in Figure 8 the following combinations of code rate and subblock size define a Pareto frontier, when the total number of LUTs of the mapped design and the maximum throughput are set as criteria: (0.83, 27), (0.75, 27), (0.66, 27), (0.83, 54) and (0.83, 81). All the other points of this figure are not Pareto efficient, because it is not possible to achieve the same or greater throughput without sacrificing additional resources.

![Fig. 9. Implementation results exploration, when throughput and the dynamic power consumption per bit are defined as criteria.](image)

According to Figure 9 the best performance, in terms of maximum throughput and lowest power consumption per input bit, is achieved for subblock size $z = 81$ bits and code rates $R = 0.75$ and $R = 0.83$. All the other points of this figure are suboptimal. For example, moving from (0.75, 54) to (0.75, 81) enables you to make throughput better off, without making the dynamic power consumption per bit worse off. whereas moving from (0.75, 81) or (0.83, 81) to any other point does not lead to any improvement in regard to these two criteria.

V. CONCLUSION

In this work, a generalized hard-wire oriented architecture for QC-LDPC encoding has been presented. Emphasis has also been given to the description of the developed Automatic VHDL Generator, which provides the user with a fast way to convert the visualized design directly to HDL code and enables rapid Design Space Exploration. The trade-offs in maximum clock frequency, throughput, resources utilization and power consumption, between architectures with different design parameters have been investigated in depth, finding that the optimal combinations of code rate and subblock size in terms of throughput and dynamic power consumption per bit are the (0.75, 81) and (0.83, 81), whereas (0.83, 27), (0.75, 27), (0.83, 54) and (0.83, 81) define a Pareto frontier, when the total number of LUTs of the mapped design and the maximum achieved throughput are set as criteria.

REFERENCES


