Implementing Double Error Correction Orthogonal Latin Squares Codes in Xilinx FPGAs

M. Demirci · P. Reviriego · J. A. Maestro

Abstract This paper studies the implementation of Double Error Correction Orthogonal Latin Squares (OLS) in Xilinx Field Programmable Gate Arrays (FPGAs). Several existing options to implement the decoder are considered and evaluated. The results show that the decoder complexity can be significantly optimized by appropriately selecting the implementation that is better suited to the internal FPGA structure. To further optimize the use of resources a version of the decoder that is tailored to the FPGA structure has also been designed. This new implementation provides significant savings compared to the existing ones. Based on these results, it seems that optimizing the ECC implementation for FPGAs can be effective and may be applied to other codes.

Keywords Error correction codes · FPGAs · Orthogonal Latin squares codes.

1 Introduction

Error Correction Codes (ECCs) are widely used to protect memories and other electronic circuits against errors [1]. Traditionally, Single Error Correction Double Error Detection (SEC-DED) codes have been used [2]. However, there is a growing interest in more advanced codes that can correct multiple errors. Advanced codes are useful to protect against Multiple Cell Upsets (MCUs) [3] and also when the error rate is large as for example when the circuits are operated at low voltages to reduce power consumption [4]. One key requirement for the ECCs that are used in memories and circuits is that the decoders have to operate at high speed to minimize the impact on the overall circuit speed. This has brought the attention of researchers to codes that can be decoded using Majority Logic (ML). With majority logic, decoding can be performed in parallel and to decode each bit only a majority vote on a number of parity check equations is needed [5]. One example of ML decodable codes are Orthogonal Latin Squares (OLS) codes [6]. OLS codes have been recently proposed to protect caches [4], interconnections [7] and memories [8]. In many cases, electronic systems are implemented using Field Programmable Gate Arrays (FPGAs). FPGAs provide a set of resources to implement arbitrary circuits. The resources are commonly structured as combinations of basic building blocks and their interconnections. To implement a circuit, a number of such blocks are configured and connected. In many FPGAs, as for example Xilinx FPGAs, one of the major building blocks are Look Up Tables (LUTs) that can be used to implement arbitrary logic functions. Implementations of ECCs in FPGAs are provided in many cases by the device manufacturers [9]. Those implementations typically map a traditional ECC decoder to the FPGA. However, the decoder structure and implementation can be tailored to match the structure and building blocks of the FPGA. This optimization can reduce the amount of FPGA resources needed to implement the ECC. In this paper, this idea is explored for Double Error Correction OLS codes. In particular its implementation for Xilinx Virtex-5 FPGAs is con-
2 Xilinx FPGA Structure

Xilinx FPGAs are built using a number of building blocks. The basic element is the Configurable Logic Block (CLB) that integrates LUTs, registers and multiplexing logic and is divided in slices [10]. The structure of one of the slices is shown in Figure 1 for a Virtex-5 FPGA (the slices also have a similar structure in Virtex-6 and Virtex-7 FPGAs). Each LUT can implement any six input logic function or two five inputs logic functions with the same inputs or two arbitrary logic functions with up to three inputs. To implement ECCs, the LUT granularity is a key factor. For example a majority voter can be implemented in a single LUT as long as it has six or less inputs. However, a majority voter with seven inputs will require several LUTs. As will be seen in the next section, taking the FPGA structure into account when designing the ECC decoder can result in significant savings compared to a direct implementation.

3 Orthogonal Latin Squares Decoders

The key parameters of an OLS code are its data block size $k$ and the number of errors that can be corrected $t$. The block size is of the form $k = m^2$ bits and to correct $t$ errors, $k = 2tm$ parity check bits are needed [6]. For example when $m = 4$, the data block size is 16 bits and if $t = 2$, the number of parity check bits is also 16. The parity check matrix of this code is shown in Figure 2. As mentioned before, the decoding of OLS codes is done using majority logic. In particular, the decoded bit can be obtained from the values of $2t$ recomputed parity check bits and the bit itself. For example for the code with $k = 16$ and $t = 2$, four parity check bits and the bit to be decoded are used. The majority vote can be implemented in two different ways. The first option is to re-compute the four parity check bits and take a majority vote among them. When there is a majority of ones, an error has occurred and the bit is corrected. The correction can be done with an xor gate. This first option is illustrated in Figure 3 (a) and named vote to correct. In the second option, the four parity check bits are partially re-computed to obtain the bit being decoded and a majority vote among those and the original bit is used to obtain the decoded bit. This is illustrated in Figure 3 (b) and named correct by voting. The two decoding options are functionally equivalent but result in very different resource utilization when implemented in a Xilinx Virtex-5 FPGA. This can be explained as in the first option the re-computation of the parity check equations is shared among all bits while in the second each partial re-computation can be used only for one bit. On the other hand, the five input majority voter of the second option can be implemented in a single LUT but the four input majority voter plus the xor gate of the first option are mapped to two independent LUTs. To minimize the resource usage, a new decoder structure has been implemented on which the four input majority gate and the xor gate are integrated in a single five input logic function. This enables the mapping to a single LUT and further reduces the cost. This third option is illustrated in Figure 3 (c) and named merged vote to correct. In the following section, the three decoder implementations are evaluated to compare the speed and resource usage.

4 Evaluation

The three different implementations discussed in the previous section have been implemented in VHDL at RTL level and mapped to a Xilinx V5LX50t-3ff1136 device. OLS codes with data block sizes of 16 and 64 bits have been considered. During synthesis effort is put on minimizing the number of LUTs while maintaining timing performance. The results in terms of number of LUTs used are summarized in Table I. It can be observed that the merged vote to correct option provides significant savings compared to the other two implementations. The number of registers is shown in Table II. In this case the merged vote to correct implementation utilizes significantly less registers than the correct by voting implementation. Compared to first implementation, there is an extra input to majority/corrector gate and combinations to be stored increase, hence the number of registers increases. The overall results clearly show the benefits of adapting the decoder to the FPGA structure. Finally, it is important to note that the savings are not only beneficial to reduce the resource usage. Using fewer LUTs, also means that the decoder is less vulnerable to soft errors in the configuration SRAM of the FPGA as there are fewer configuration bits involved in the decoder implementation. This has been previously noted for other ECC implementations on FPGAs [11].
Implementing Double Error Correction Orthogonal Latin Squares Codes in Xilinx FPGAs

Fig. 1 Block diagram of a Xilinx FPGA slice

Fig. 2 Parity check matrix for the OLS code with $k = 16$ and $t = 2$
Fig. 3 OLS Decoder Implementations

Table 1 Number of LUTs required

<table>
<thead>
<tr>
<th>k</th>
<th>Vote to correct</th>
<th>Correct by voting</th>
<th>Merged vote to correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>49</td>
<td>80</td>
<td>33</td>
</tr>
<tr>
<td>64</td>
<td>193</td>
<td>394</td>
<td>129</td>
</tr>
</tbody>
</table>

Table 2 Number of registers required

<table>
<thead>
<tr>
<th>k</th>
<th>Vote to correct</th>
<th>Correct by voting</th>
<th>Merged vote to correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>48</td>
<td>95</td>
<td>64</td>
</tr>
<tr>
<td>64</td>
<td>128</td>
<td>383</td>
<td>192</td>
</tr>
</tbody>
</table>

5 Conclusions

In this paper, the implementation of double error correcting Orthogonal Latin Square codes in FPGAs has been studied. Two existing decoder implementations have been evaluated and a new implementation has been proposed to optimize the mapping to the FPGA structure. The results show that by adapting the decoder to the FPGA structure significant savings can be obtained in the number of LUTs required to implement the decoders.

References