International Conference on Information and Communication Technologies (ICICT 2014)

Synthesis of Reconfigurable Video Compression Modules in Virtex FPGAs for Multiple Fault Repair Mechanism

Neena Baby\textsuperscript{a,\ast}, Pradeep C\textsuperscript{b}, Saranya R\textsuperscript{c}, Radhakrishnan R\textsuperscript{d}

\textsuperscript{a,b}\textsuperscript{c}Dept. Electronics and Communication Engineering, Saintgits College of Engineering, Kottayam-686532, India
\textsuperscript{d}Vidhya Mandhir Institute of Technology, Perundurai, Tamilnadu, India

Abstract

This paper proposes a self-repairable video compression module that can be used for space applications. A brief idea of the self-repair technique that is capable of handling multiple faults and the various steps involved in designing the video compression modules is discussed in this paper. The different steps involved in compression are designed using Verilog HDL. The design of compression steps as separate reconfigurable modules is simulated and synthesized using Xilinx ISE 14.2. The design is also synthesized using various Virtex FPGA devices to find out which device will save more logic resources and time on implementing the technique.

© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

Keywords: FPGA; video compression; radiation; self-repair

1. Introduction

FPGAs are semiconductor logic devices. They have been used in space applications for a long period of time. The cost and time needed to implement FPGAs is decreasing. Also they are simple, flexible and reprogrammable. These are the major factors leading to their success in space and avionic applications. As the steps in manufacturing can be neglected, tooling for FPGAs is less complex and cheaper in comparison with ASICs.

* Corresponding author. Tel.: +919747469091.
E-mail address: neenab90@gmail.com
FPGAs have a crucial responsibility in space applications, particularly in geographical and weather forecasting processes where the significant steps such as image and video compression can be performed in orbit. Due to the flexible nature of FPGAs, hardware implemented image and video compression algorithms can be exchanged from the ground station on demand. So FPGAs can be used for these applications. The reconfigurable feature of FPGAs can be used in space applications and in satellites which are in space or in orbit.

In space-flight application, the electronic components require high reliability. However, a major reliability threat arises from the radiation that exists in space. The radiation effects include Total Ionizing Dose (TID), Single Event Effects (SEE), susceptibility to electromigration and thermal runaway. The scaling of CMOS has made the feature size in nanoscales and as a result, the radiation effects on the chips are increasing these days. So exposure to radiation will adversely affect the chips in nanoscale.

The aim of this work is to develop an approach that allows the designs to be automatically reliable for space applications. Several existing technologies aim at alleviating the problems that occur in space applications. But in certain cases, faults propagate through the design and make the system fails in critical space applications. As a result, novel techniques are required to handle the effects of radiation in nanoscale designs. These investigations resulted in the development of fault tolerance approach for reliable design.

The fault tolerant systems can be designed in several steps including fault detection, fault confinement, fault recovery, fault treatment and continued system service. The proposed work concentrates on the fault recovery stage. It deals a technique which is capable of removing an error from an MPEG-2 video compression module used in space applications. Whenever a fault occurs in any of the modules in video compression, a self-repair technique is developed so that the system can be retained for a longer duration. In this paper, different steps of compression are designed as separate reconfigurable modules and device utilization of these modules on various Virtex FPGA devices is checked to see which device will save more logic resources and time on implementing the technique.

The rest of the paper is organized as follows. Section 2 deals with the related research works in this area. The design of reconfigurable modules and the proposed self-repair technique for obtaining a self-repairable MPEG-2 video compression module is presented in Section 3. The result and analysis is given in Section 4 and the paper concludes with mentioning the future scope in section 5.

2. Related works

A pipelined architecture of Two Dimensional Discrete Cosine Transform (2D-DCT) with quantization and zigzag arrangement for JPEG image compression is proposed\(^1\). 2D-DCT Separability property is used for the calculation of 2D-DCT. A transpose buffer is used for dividing the architecture into two One Dimensional Discrete Cosine Transform (1D-DCT) calculations and quantization is performed by division operation. The sequential operation of DCT saves logic utilization in FPGA and the method is fast and less complex. A simple Finite State Machine (FSM) architecture for the 2D-DCT computation, quantization and zigzag ordering for JPEG image compression is presented\(^2\). A simulation study of the normalization matrix suitable for hardware simplification and another recommended by JPEG shows that the former can be used in JPEG baseline image compression. The implementation of 1D-DCT is done for the compressed Distributed Arithmetic (DA) algorithm with shifting performed by division operator. The results show a considerable amount of savings in hardware when quantization is carried out by wired shifting operation.

K. Papadimitriou et al. proposed a concept for measuring the reconfiguration time\(^3\). The experimental setup includes XUPV2P platform with a Virtex-2 Pro FPGA, a board with LEDs, an Agilent 1680A logic analyzer and a PC. H. Tan and R. F. Demark published a paper on Multilayer Run-time Reconfiguration Architecture (MRRA) for independent run-time Partial Reconfiguration (PR) of FPGAs\(^4\). The hardware model of MRRA is developed for a Xilinx Virtex II Pro VP7 FPGA on an Avnet Virtex II Pro development board. P. Patel established the use of Xilinx Virtex-5 FPGAs in embedded applications\(^5\). The strength of Xilinx FPGAs and the related development tools is also discussed.

N. Abel et al. presented a paper on Dynamic Partial Reconfiguration (DPR) and DPR tools\(^6\). Early-access routing tools are supported by Xilinx and PlanAhead provides a platform for PR. W. Lie and W. Feng-yam proposed a technique to reduce the configuration time and to save memory using PR system\(^7\). The advantages of the most recent PR design flow are also discussed. S. Corbetta et al. presented the partial bitstream relocation activity\(^8\). The
method has been validated over different FPGAs such as Virtex II Pro, Virtex 4 and Virtex 5, so the method can be customized to suit various constraints associated with the target architectures.

Additional hardware components are essential in hardware redundancy techniques for masking the presence of Single Event Upsets (SEUs) or correcting them. Fault detection and masking can be achieved in FPGAs by triplicating the circuit implemented on FPGA. A fault tolerant technique called Triple Modular Redundancy (TMR) can be used in integrated circuits for tolerating against the errors. TMR uses three identical logic blocks that performs the same task. Voters are used to compare the corresponding outputs as shown in Fig. 1. Area overhead and high implementation cost are the drawbacks of this technique.

Self-propagation and self-healing characteristics of biological organisms were used in the design of digital systems by Lala. The architecture consists of router, functional and spare cells. Two spare cells surround each functional cell. Whenever a functional cell is found to be faulty, it is replaced by a suitable spare. A decision making circuitry is also required in the spare and router cells. This is used to decide the suitable spare cell to replace the faulty cell.

Kim et al. proposed a self-repairing architecture for fast fault recovery. This is done with the efficient utilization of limited number of resources. The architecture was inspired from the concept of paralogous genes. A working module consists of 16 working cells, 16 redundant cells and 4 stem cells. Each working cell has its own fault detection circuit. The normal operation is immediately restored by replacing a faulty working cell by its redundancy. Each functional cell has a spare and the inputs are pre-routed to both. This still requires an external router to decide whether the original cell or the spare cell needs to be active. In case of a permanent fault, the stem cells which are initially empty become redundant for any working cell.

3. Proposed work

The proposed work deals with the development of a self-repairable video compression module that can be used for space applications. For obtaining a self-repairable video compression module, the different steps in the compression procedure are designed as separate reconfigurable modules and a self-repair technique is developed so that the module can be sustained for a longer duration. The proposed technique can handle multiple faults, i.e., transient as well as permanent faults. The technique handles unlimited number of transient faults and it makes use of spare cells and least priority modules to handle permanent faults.

3.1. Design of reconfigurable modules for self-repairable MPEG-2 video compression module

Video compression is referred as the process of dropping the amount of data to represent a video. This is done inorder to meet a particular bit rate requirement. Here, the reconstructed video quality needs to meet the requirement for a certain application and the application must be capable of affording the involved computational complexity. It is found that video compression is necessary in most of the important applications as the amount of data involved in these applications is very huge and it usually exceeds today’s hardware capability despite of the rapid advancements in modern industries. Fig. 2 (a) shows the functionality of video compression.

MPEG-2 video compression was defined and standardized by the Motion Picture Expert Group in 1994 and is used in DVDs, digital television and numerous other digital video devices. The sequence of steps in MPEG-2 video compression encoding is shown in Fig. 2 (b).
3.1.1. DCT

The DCT transforms information from the spatial domain to the frequency domain. DCT is a special case of Fourier transform. An input signal is represented in a series of sine and cosine terms in Fourier transform. However in DCT, the sine components are eliminated\(^{13}\). Here, 1D-DCT is calculated twice to attain the 2D-DCT. 8x8 blocks of pixels are taken for calculating 2D-DCT. The resulting set of data is an 8x8 block of frequency space components known as basic functions. The average frequency value of the entire block is represented by the first element at row 0 and column 0 and it is known as the DC term. The remaining 63 terms represent the spatial frequencies that make up the input pixel block by scaling the cosine terms within the series and are known as AC components.

A black and white image can be represented in pixel values from 0 to 255 in steps of 1. Here, pure white and pure black is represented by 0 and 255 respectively. Thus 256 shades of gray can be used to exactly represent a photo. Hundreds and thousands of 8x8 blocks of pixels is comprised in an image and the method for representing one 8x8 block is similarly done to all other 8x8 blocks of pixels. Consider the upper left-hand corner of an image and a block of pixel values is taken from there.

The equation for 1D-DCT for \(X[i]\) be the input is given by:

\[
F(u) = \frac{1}{2} C(u) \sum_{i=0}^{7} X[i] \cos \left( \frac{\Pi (2i + 1) u}{16} \right)
\]

(1)

where \(C(u) = \begin{cases} 
1, & u = 0 \\
\frac{1}{\sqrt{2}}, & \text{otherwise}
\end{cases}\)

3.1.2. Quantization

Quantization is essential for compressing the 8x8 block of DCT coefficients\(^{13}\). Exact quantization matrices are chosen appropriately to achieve different levels of video compression and quality. The quality level of an image ranges from 1 to 100. Here, 1 gives the lowest quality image and highest compression whereas 100 give the highest quality image and lowest compression. The quantization matrix is obtained from several experiments and the matrix results in an image that has high compression and exceptional quality at a quality level of 50.

Each element in the quantization matrix divides the corresponding element in the 8x8 block of DCT coefficients and it is then rounded to the nearest integer value to carry out quantization. The lower frequency components are represented by the coefficients near the upper-left corner and the higher frequency components that have been discarded are represented by zeros. The image is reconstructed using the non-zero components.

The quantization matrix \(Q_{50}\) used here is shown below.
3.1.3. Run-Length Encoding (RLE)

The next level of compression is now done on the quantized matrix. After quantization, the maximum number of coefficients is equal to zeros. The quantized coefficients are now encoded in zigzag order. The image components are arranged in zigzag order and the identical frequencies are assembled together to perform RLE. In RLE, a single data value and count is used to represent the same data value that occurs in many consecutive data elements. This is very useful for the data that contains the same value in successive elements. RLE attains good compression only if there are many similar successive data elements.

\[ Q_{so} = \begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 100 & 103 & 99
\end{bmatrix} \]

3.1.4. Huffman coding

From Shannon’s Source Coding Theory, an average code length close to the entropy of the source can be used for coding a source. In 1952, D.A. Huffman invented a coding technique to find the shortest possible average code length, provided the source symbol set and associated probability of occurrence of the symbols are given. Huffman coding is a method of dropping the number of bits required to signify a set of values, by generating shorter encodings for the repeatedly occurring values, and longer encodings for the less repeated value. Codewords with the same length are used to represent the two least repeatedly occurring symbols where only the LSB will be different. Codes generated using these coding techniques are known as Huffman codes.

Huffman coding attains good compression when some numbers occur much more repeatedly than other numbers in the sequence of numbers to be encoded. Fortunately, this is indeed the case after DCT, quantization and run-length tasks are carried out on a block of a frame. For example, there may be plenty of 0s, 1s, 2s, etc., and less occurrences of higher numbers.

3.2. Self-repair technique for MPEG-2 video compression module

A self-repair technique is developed to handle multiple faults such as transient and permanent faults. The technique makes use of the availability of spare cells. It can handle unlimited number of transient faults (temporary changes in the logic circuit properties), but the number of availability of spare cells determine the capacity of handling permanent faults (alter the implemented logic functions permanently), i.e., if a permanent fault occurs in the absence of a spare cell, the system fails. This condition can be overcome by making use of the least priority module. If a permanent fault occurs in the absence of a spare, then the functionality of the faulty module is handed over to the least priority module. This can be continued by making use of all the least priority modules. So the reliability of the entire system can be improved.

For obtaining a self-repairable MPEG-2 video compression module, the different steps in the compression procedure are designed as separate reconfigurable modules as shown in Fig. 3 and the above mentioned steps of repair are applied to the compression module. Huffman coding has got the least priority in the MPEG-2 video compression module. So if a permanent fault occurs in the absence of a spare cell, the functionality of the Huffman coding is neglected and the same one will take the functionality of the faulty module.

A model of sustaining MPEG-2 video compression module for a longer duration is shown in this section. The method can be employed in a similar way to all other compression schemes and can be extended further by using better compression schemes.
4. Result and Analysis

The design of the video compression system is done by taking the pixel values in the form of an 8x8 matrix. These pixel values are given as the input of the DCT block. As shown in Fig. 4 (a), pixel 1 to pixel 8 are the inputs and dct1 to dct8 are the outputs of the DCT block. Fig. 4 (b) shows the simulation waveform of the quantization block. The output of the DCT block is given as the input of the quantization block. The inputs and outputs of the quantization block are dct1 to dct8 and quant1 to quant8 respectively. It is then given as the input of the RLE block. As shown in Fig. 4 (c), quant1 to quant8 and rle1 to rle8 are the inputs and outputs of the RLE block respectively. The final compressed bits are obtained from the Huffman encoder. As shown in Fig. 4 (d), rle1 to rle7 are the inputs and huff_1 to huff_7 are the outputs of the Huffman encoder. The variable ‘huff’ indicates the overall compressed bits. It is clear from the result that the final output bits are greatly reduced.

Table 1 shows the device utilization of MPEG-2 video compression module on Virtex-5, Virtex-6 and Virtex-7 devices. Virtex-5 occupies 4409 slices and 13182 LUTs, Virtex-6 occupies 2499 slices and 7960 LUTs and Virtex-7 occupies 1872 slices and 5647 LUTs. Also Virtex-5, Virtex-6 and Virtex-7 take 44.294 ns, 37.473 ns and 20.103 ns respectively to obtain the final compressed bits. However on implementing the proposed technique using above devices, i.e., neglecting the functionality of Huffman coding module and making it to function in place of the faulty module in case of a permanent fault with the absence of a spare, Virtex-5 saves more logic resources and time compared to other devices. Virtex-5 saves 55 slices, 64 LUTs and it reduces the overall delay by 4.276 ns. However, Virtex-6 and Virtex-7 saves less number of logic resources and time in comparison with Virtex-5. The comparison of the saved logic resources and time on implementing the technique using various devices is shown in Fig. 5.

Table 1. Device utilization of MPEG-2 video compression module on various devices.

<table>
<thead>
<tr>
<th>Device</th>
<th>Modules</th>
<th>Slice occupied</th>
<th>LUTs</th>
<th>Number of IOBs</th>
<th>Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtex-5 (XC5VLX110T)</td>
<td>DCT</td>
<td>4233</td>
<td>12984</td>
<td>512</td>
<td>30.598</td>
</tr>
<tr>
<td></td>
<td>Quantization</td>
<td>61</td>
<td>70</td>
<td>314</td>
<td>5.144</td>
</tr>
<tr>
<td></td>
<td>RLE</td>
<td>60</td>
<td>64</td>
<td>256</td>
<td>4.276</td>
</tr>
<tr>
<td></td>
<td>Huffman coding</td>
<td>55</td>
<td>64</td>
<td>192</td>
<td>4.276</td>
</tr>
<tr>
<td>Virtex-6 (XC6VLX240T)</td>
<td>DCT</td>
<td>2356</td>
<td>7812</td>
<td>512</td>
<td>25.663</td>
</tr>
<tr>
<td></td>
<td>Quantization</td>
<td>60</td>
<td>68</td>
<td>314</td>
<td>4.418</td>
</tr>
<tr>
<td></td>
<td>RLE</td>
<td>47</td>
<td>45</td>
<td>256</td>
<td>3.696</td>
</tr>
<tr>
<td></td>
<td>Huffman coding</td>
<td>46</td>
<td>45</td>
<td>192</td>
<td>3.696</td>
</tr>
<tr>
<td>Virtex-7 (XC72000T)</td>
<td>DCT</td>
<td>1775</td>
<td>5542</td>
<td>512</td>
<td>17.133</td>
</tr>
<tr>
<td></td>
<td>Quantization</td>
<td>34</td>
<td>41</td>
<td>314</td>
<td>1.306</td>
</tr>
<tr>
<td></td>
<td>RLE</td>
<td>32</td>
<td>32</td>
<td>256</td>
<td>0.832</td>
</tr>
<tr>
<td></td>
<td>Huffman coding</td>
<td>31</td>
<td>32</td>
<td>192</td>
<td>0.832</td>
</tr>
</tbody>
</table>
Fig. 4. Simulation waveforms for (a) DCT; (b) quantization; (c) RLE; (d) huffman coding.

Fig. 5. Comparison of saved logic resources and time on implementing the technique using various devices (a) slices; (b) LUTs; (c) delay.
5. Conclusion and Future work

The design of an efficient and reliable FPGA based MPEG-2 video compression approach as separate reconfigurable modules to meet self-repair capability are discussed in this paper. Whenever a fault occurs in any of these modules, a self-repair technique capable of handling multiple faults is also proposed so that the module can be retained for a long period of time. The design is done using Verilog HDL, simulated and synthesized using Xilinx ISE 14.2. The synthesis of the video compression module is done using various Virtex FPGA devices and it is clear from the result that on implementing the technique, Virtex-5 saves more logic resources and time in comparison with Virtex-6 and Virtex-7. However, the amount of compression may decrease on implementing the technique.

The future goal is the development of a self-repair technique for MPEG-2 video compression module that can satisfy multiple objectives such as less area and minimum routing overhead and its implementation using PlanAhead 14.2 on Virtex-5 devices which supports partial reconfiguration.

References


