Implementation of improved error trapping decoder for multiple error correcting cyclic codes in a soft core processor

S B Gayen1*, B Dam2 and D Patranabis2

1Radar Data Processing Division, Research Centre Imarat, DRDO, Vignyanalancha (P O), Hyderabad 500 069
2Department of Instrumentation and Electronics Engineering, Jadavpur University, Kolkata

Received 10 March 2008; revised 03 June 2008; accepted 09 June 2008

This paper presents implementation of Tadao Kasami decoding algorithm for improved error trapping decoding and firmware realization details. Algorithm has been applied to cyclic (n, k) codes [Golay (23, 12) and BCH (31, 16)] for triple error correction. Microblaze 32-bit soft core processor was used with a Xilinx Spartan3 FPGA. To test decoder functionality, a test procedure with all possible error combination is devised. Profiling of execution time with no error and with error is presented along with hardware resource utilization of FPGA.

Keywords: Error trapping decoder, Kasami decoding algorithm, Soft core processor

Introduction

Error trapping decoder (ETD) using combinational logic, microcontroller and standard processor is being used in most new data communication systems, microwave satellite communication systems and digital data storage medium. Using Field Programmable Gate Array (FPGA) technology and soft core processor, decoder and encoder can be developed fast for different coding algorithms. Generally, decoding algorithms (in C/C++) require only a few kilobytes of memory size; hence target code can reside inside Block RAM (BRAM) of FPGA. This removes necessity of additional external flash memory for processor, thereby reducing cost and complexity of design. Glue logic and IP core etc. can be used along with processor within FPGA. This provides a tremendous advantage in system design, as designer if not proficient in VHDL or Verilog HDL languages can take advantage of high level coding languages. Meggitt1 decoders can be used to decode any cyclic code. Variation of Meggitt decoding is called error trapping decoding2-4, which employs very simple combinational logic circuits for error detection and correction. When decoder has to handle large code length ‘n’ and error correcting capability ‘t’, and is required to have burst error correction capability, improved ETD mechanism is used.

This study presents implementation of an improved ETD devised by Kasami5, which is most suitable for System ON Programmable Chip (SOPC) applications.

Proposed Decoder

Basic features of Microblaze Processor

MicroBlaze6 embedded soft core is a 32 bit Reduced Instruction Set Computer (RISC) optimized for implementation in Xilinx FPGA devices. Both instruction and data interfaces of Microblaze are 32 bit wide and use big endian bit reversed format. There are 32 general purpose registers, 32-bit instructions with three operands and two addressing modes. This soft core, by virtue of massively parallel General Purpose Input and Output (GPIO) configurability and access to various embedded cores, provide new alternatives for creating high performance applications.

Encoding and Decoding of Codes

Cyclic codes4 are generated based on a generator polynomial, g(x), which is used while decoding received message. The g(x) for Golay code1,3,6 is \( x^{11} + x^9 + x^7 + x^5 + x + 1 \) and for BCH code2-4 is \( x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \). Codes are generated using standard cyclic code encoding algorithm2-4 and implemented in C language. Kasami algorithms5 for decoding multiple error correcting cyclic codes is used to decode triple error correcting Golay and BCH code.
This decoding scheme is simple and appropriate for Golay and BCH codes, where minimal Hamming distance \(d_{\text{min}} \geq 7\) is relatively large with respect to code length.

**Design and Implementation**

A digital system has been realized to demonstrate implementation and evaluate performance of proposed codes (Fig. 1). The system uses Microblaze\(^6\) soft core (version v 4.00a) as main processor along with external memory, communication port (RS232), debugging and programming interfaces. Soft core processor is targeted to a Xilinx XC3S4000, Spartan3 FPGA. Processor is configured in Xilinx Platform Studio\(^7,8\) (XPS) using base system builder wizard with a system clock of 100 MHz. After synthesis and implementation, a bit file is generated and programmed into platform flash of FPGA through Xilinx parallel IV cable using JTAG port. Application programs are compiled in Software Development Kit\(^7,8\) (SDK) and generated hex file is downloaded into processor external flash memory through XPS flash programmer tools. With a system reset, program is executed and results are logged in two files to validate decoding algorithms. Processed data are read into PC offline through a communication port (RS 232). Algorithms for decoding BCH (31, 16) and Golay (23, 12) are identical and are based on Kasami decoder but generator polynomials and covering polynomials are different. A unified approach has been adopted to test and evaluate both the decoders one by one in totality.

**Flow Chart for Evaluation**

Flow chart (Fig. 2) depicts steps required to evaluate decoding program. Code length \(n\) and information \(k\) are different for BCH and Golay codes, and as such loop depth will be more for BCH code. Possible number of valid code vectors for Golay and BCH codes are 4096 and 65536 respectively. In flow chart, \(i\) is encoder input for Golay (0-4095) BCH (0-65535) codes. Encoder output fed to decoder with and without error. Outputs of decoder are stored in two different files, one for correctly decoded values and other for incorrectly decoded values. These files are retrieved later to analyze performance of decoder.

**Error Generation and Testing**

It is mandatory to evaluate decoders for all possible combination of errors, which can be random (1-error, 2-error, 3-error and burst error). Inputs to error generator (Fig. 3) are encoded code vectors (CV). In three loops (Fig. 3) all possible errors have been generated and are stored in 3 different files. There are 23 1-error, 253 2-error and 1771 3-error patterns possible for Golay code. Similarly, 31 1-error, 465 2-error and 4995 3-error patterns are possible for BCH code. These error files are used as erroneous data input to evaluation algorithm (Fig. 2). Three error mask are used to generate the errors. Mask bits are shifted from LSB to MSB in three nested loop.
Generated erroneous code vector are stored in three error file (e1, e2 and e3).

**FPGA Resource Utilization**

MicroBlaze processor is configured with 32 Kb local memory (BRAM), XMD debugger, RS232 port and without cache memory in a Xilinx Spartan 3 FPGA (XC3S4000). Processor uses a very small percentage of FPGA resource (Table 1). So, plenty of resource are available for other logic and core utilization. Thus, this implementation provides a suitable platform for SOPC applications incorporating ETDs.

**Execution Time**

Measurement of program execution time on target hardware is done with a 32 bit timer counter in count up mode as follows: i) Initialize a timer counter (TC) of processor at the start of program with precision of system
Fig. 3—Flowchart for error generation

- $i_{\text{max}} = 23$ for Golay code and 31 for BCD code
- $j_{\text{max}} = 22$ for Golay code and 30 for BCD code
- $k_{\text{max}} = 21$ for Golay code and 29 for BCD code
- $e_{\text{max}} = \text{errormask1}$, $e_{1} = \text{error1}$
- $e_{2} = \text{errormask2}$, $e_{2} = \text{error2}$
- $e_{3} = \text{errormask3}$, $e_{3} = \text{error3}$
clock (SC) [Start_TC = 0, System clock =100 MHz (10 ns)]; ii) Stop timer counter at the end of program execution (Stop_TC); and iii) Timer counter values are listed as number of clock cycles. So,

\[
\text{Measured execution time} = [(\text{Stop_TC} – \text{Start_TC}) \times SC\text{time}]
\]

As the position of error bit is random within code vector, execution time will deviate from one error bit position to another error bit position (Table 2). For all errors, which are located within \((n-k)\) position of code vector, will match the syndrome\(^2\) \(\text{directly with least execution time. If errors are located above (n-k) position of code vector then execution time increases based on error distribution over code vector.}

Conclusions

SOPC implementation of decoders provides a configurable and platform independent hardware, which can be used for implementation and validation of other category of block code and convolutional code encoder & decoder. Maximum program execution time is \(~500\) μs, but there is a room for improvement with optimizing decoding algorithms and operating processor at higher system clock. Present hardware can be used for moderately high speed communication system applications, like data link receiver, wireless data acquisition system etc. Hardware resource utilization of FPGA is very less; thereby hardware acceleration of the desired system is possible with spare resource of FPGA. In future, this work can be done with other soft core processor (Power PC 705, NIOS and ARM 9) and a performance comparison can be drawn to choose optimum processor most suitable for this application.

Table 1 — FPGA resource utilization

<table>
<thead>
<tr>
<th>Logic utilization</th>
<th>Used</th>
<th>Available, no.</th>
<th>Utilization, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice flip flops</td>
<td>773</td>
<td>26624</td>
<td>2.9</td>
</tr>
<tr>
<td>4 Input LUTs</td>
<td>1709</td>
<td>26624</td>
<td>6.4</td>
</tr>
<tr>
<td>Block RAM</td>
<td>64 Kb</td>
<td>1728 Kb</td>
<td>3.7</td>
</tr>
<tr>
<td>Multiplier 18 x 18</td>
<td>3</td>
<td>32</td>
<td>9</td>
</tr>
<tr>
<td>DCMs</td>
<td>1</td>
<td>4</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 2 — Execution time for a given clock and code

<table>
<thead>
<tr>
<th>Decoder</th>
<th>System clock</th>
<th>No error</th>
<th>1 error</th>
<th>2 error</th>
<th>3 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Golay (23, 12) 100MHz</td>
<td>29.36</td>
<td>197.09</td>
<td>465.17</td>
<td>469.23</td>
<td></td>
</tr>
<tr>
<td>BCH (31, 16) 100MHz</td>
<td>41.70</td>
<td>213.50</td>
<td>482.91</td>
<td>486.21</td>
<td></td>
</tr>
</tbody>
</table>

References

7. www.xilinx.com