**Multiple Table Lookup Implementation of Error Correction on an FPGA**

Amila Akagić, Hideharu Amano  
Dept. of ICS, KEIO University, 3-14-1 Hiyoshi, Yokohama, 223-8522 Japan  
Email: \{amilahunga\}@am.ics.keio.ac.jp

**ABSTRACT**

In recent studies substantial number of research papers explored possibility of accelerating CRC codes as a way of gaining significant performance increase while processing a protocol. Most software implementation depend on a speed and architectural improvements of general purpose processors. It has been reported that a single CPU core still can’t achieve 10Gbps rates when receiving and processing bulk data (>2KB) [1]. Widely used Sarwate algorithm is being replaced by a newly proposed Slicing-by-8 algorithm, with highest reported throughput of 3.6Gbps on a 1.7GHz Pentium M processor [2]. CRC algorithms are highly sequential, thus it’s not known how they will take advantage of multicore processors. We believe that in order to achieve higher throughput (ex. 10Gbps or 40Gbps), new technologies have to be considered.

Based on a [2] framework, we design and implement Slicing-by-{4, 8, 16, 32} algorithms, on Xilinx Virtex 5 LX30 device (5vlx30ff676), at speed grade of -3. Algorithms read and process 32, 64, 128 and 256 of input data at a time, respectively. They are based on two principles of modulo-2 arithmetic, bit slicing and bit replacement, so they use different number of lookup tables with precomputed CRC values. Block diagram of Slicing-by-4 is displayed on Fig. 1. Other implementations differ only by a way we access memory and by a number of lookup tables used. We present results in the Table I. Maximum throughput attained is 79.86Gbps while processing 256 bit at a time. Other implementations achieve following maximum throughput: 12.66Gbps for processing 32 bits, 24.93Gbps for processing 64 bits and 41.99Gbps for processing 128 bits at a time.

Input buffer for Slicing-by-{4, 8, 16} is implemented to read 128 bits and Slicing-by-32 reads 256 bits from a bus. Slicing-by-4 algorithm reads 32 bits at a time, so our implementation uses a mux to make a selection of four possible words from the input buffer (Fig. 1). This value is stored in a register that is used as the first operand for the first XOR circuit. Second operand depends on a current iteration step. Initial value, defined by a CRC standard, is used only in the first iteration, and other iterations use previously calculated CRC value. Output from the second XOR circuit is then sliced into a number of 8 bit slices, used to access tables (T1-4 on Fig. 1). Lookup tables are implemented as 256x32-bit ROM modules. Outputs from these tables are XORed and the results are saved into another register. This value is used in the next iteration. In final stage, last CRC value is XORed with a final value, which is also specified by a CRC standard used.

FPGAs can access lookup tables in parallel, and this is how our implementations were able to maintain increase in throughput. Our implementation can easily be modified to implement various 32bit generator polynomials, only by changing content of tables. Our circuit can be integrated with an existing network interface and process CRC as packets arrive, which will decrease interrupt processing cost of a CPU as well as protocol processing latency.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Slice registers</th>
<th>Slice LUTs</th>
<th>IOBs</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slicing-by-4</td>
<td>236 (1%)</td>
<td>259 (1%)</td>
<td>165 (41%)</td>
<td>404.99</td>
</tr>
<tr>
<td>Slicing-by-8</td>
<td>266 (1%)</td>
<td>437 (2%)</td>
<td>165 (41%)</td>
<td>398.82</td>
</tr>
<tr>
<td>Slicing-by-16</td>
<td>327 (1%)</td>
<td>663 (3%)</td>
<td>165 (41%)</td>
<td>335.82</td>
</tr>
<tr>
<td>Slicing-by-32</td>
<td>583 (3%)</td>
<td>1286 (6%)</td>
<td>293 (73%)</td>
<td>319.44</td>
</tr>
</tbody>
</table>

**Fig. 1.** Block diagram of our design for Slicing-by-4 algorithm.

**Table I**

The synthesis results for Slicing-by-{4, 8, 16, 32} algorithms for the Xilinx Virtex 5 LX30.

**REFERENCES**
