ACS Implementation of the Lempel-Ziv Compression

Robert Wei-Je Huang
RATS / EE385A
Stanford University
February 1, 1999

Outline
- Introduction
- LZ-based compression
- FPGA implementation
- Potential on error-checking ability
- Reference

Introduction
- Advantage of ACS
- Current Research
- Target Application
  - Robotics / Control
  - Compression / Encryption
  - Signal Processing: FFT

LZ-Based Compression
- Universal compression algorithm
- Dynamic dictionary: uses previously seen source phrases to construct the dictionary
- Encoding: replaces variable-length phrases in the source file with pointers into the dictionary

First Version: LZ-77 Algorithm
- Find the maximum-length matching phrase between the input sliding window and the current dictionary array
- Update the dictionary once a matching operation is done

Other Versions
- LZSS
- LZ78
- LZW - popular compression standards for modern communication and data storage
  - e.g. GIF file format, CCITT V.42bis modem standard
- LZT
- LZC

Diagram:
- Input Sliding Window
- Comparator Array
- Dictionary Array
- Output = (Index, Length, Character)
ACS Implementation of the Lempel-Ziv Compression

LZW Algorithm
- Dictionary is always preloaded
- Dictionary cell contains the first symbol beyond the max-length matching string and the pointer to that string in the dictionary
  - Example: “computer” = “r” + the index of “compute” in the dictionary
- No more updates after the dictionary is full
- Output the index of the max-matching phrase in the dictionary only

Performance Comparison: LZ-77
- Advantage
  - Better temporal locality
  - Decoding is very simple
- Disadvantage
  - Poor compression ratio for small input source
- Compression ratio for 4K-entry dictionary:

<table>
<thead>
<tr>
<th>Type</th>
<th>LZ-77</th>
<th>LZ-78</th>
<th>LZW</th>
<th>LZC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio</td>
<td>3.94</td>
<td>4.13</td>
<td>4.74</td>
<td>4.26</td>
</tr>
<tr>
<td>(bits/char.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Performance Comparison: LZW
- Advantage
  - Less redundant information in the output
  - No need to wait for initializing the dictionary both in encoding and decoding
- Disadvantage
  - Compression ratio is highly dependent on the dictionary size

<table>
<thead>
<tr>
<th>Size in entries</th>
<th>512</th>
<th>1k</th>
<th>2k</th>
<th>4k</th>
<th>8k</th>
<th>16k</th>
<th>32k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio (bits/char.)</td>
<td>6.15</td>
<td>5.45</td>
<td>5.05</td>
<td>4.74</td>
<td>4.32</td>
<td>4.10</td>
<td>4.05</td>
</tr>
</tbody>
</table>

Dictionary Management
- Software approach: trie, hashed table...
- Hardware approach: Content Addressable Memory (CAM), Systolic Array
- CAM approach: Fast and easy way to achieve the realtime encoding
- Systolic Array: More hardware-efficient, but the overhead and the resulting speed may be an issue

FPGA Implementation of LZW using CAM approach
- Verilog coding and Quickturn M250 synthesis/emulation
- Problem: synthesis tools generally do poor jobs on memory arrays
- Quickturn M250 supports LM-based RAM up to 2KB. Need external cards for more memory or special types of circuitry like CAM

Result of Quickturn compilation
- Fail to import the design if dictionary size >= 2K entries due to memory insufficiency
- Number of FPGA chips required in Quickturn emulation (Quickturn M250 has 80 FPGA chips):

<table>
<thead>
<tr>
<th>Dictionary Size</th>
<th>2-port CAM</th>
<th>1-port CAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>512-entry</td>
<td>106</td>
<td>48</td>
</tr>
<tr>
<td>1k-entry</td>
<td>216</td>
<td>166</td>
</tr>
</tbody>
</table>
ACS Implementation of the Lempel-Ziv Compression

Systolic Array Approach on LZ-77

Potential on Error-Checking Ability of LZ-77 approach

Summary

- FPGA implementation of LZ-based compression is limited by the dictionary size
- LZ-77 is more feasible
- Decoding process can be multi-threaded with the encoding to improve the error-checking ability

Reference