WP152 (v1.0) September 25, 2001

Xilinx FPGA Configuration Data Compression and Decompression

By: Arthur Khu

This document provides a brief description of the Xilinx bitstream compression algorithm based on the LZ77 scheme. FPGA configuration files can be compressed by Xilinx-developed software to reduce memory storage requirements. Compressed configuration files can be stored in a high-density System ACE™ MPM FPGA configuration controller. The System ACE MPM controller decompresses the files and shifts the original configuration data to the target FPGAs.

Overview of LZ77

LZ77 is a dictionary-based text compression scheme developed by Abraham Lempel and Jacob Ziv in 1977. The scheme works by defining a fixed-size dictionary to hold bytes from an input source (e.g., a file), and then referring to the dictionary when compressing the remainder of the input source to find existing patterns. If a pattern in the input source is already in the dictionary, this pattern is replaced with a reference to the position in the dictionary and the length of the pattern.

As compression progresses, the dictionary is updated by shifting in more bytes from the input source, subsequently forcing earlier entries out. The dictionary is sometimes referred to as a sliding window because it contains a constantly updated snapshot of the input stream.

Decompression is the inverse of the compression process. The same fixed-size dictionary (or sliding window) is used to hold uncompressed data. When a reference to the dictionary is encountered, the decompressor simply copies the specified number of bytes from the dictionary, shifts these bytes into the same dictionary, and then continues processing the rest of the compressed data stream.

Compression Example

Assume a 20-byte dictionary / sliding window and an 8-byte look-ahead window implemented as First-In-First-Out (FIFO) data structures.

```
     Sliding Window    | Look-Ahead Window
AABBAA AEAAA EEFFFAAAG |   AEAAACEE
           1111111111
01234567890123456789      01234567 ← sliding window position
```

The sliding window is loaded with the first 20 bytes from the input source, with symbol or data byte A in window position 0 and symbol G in window position 19. The remainder of the input file consists of bytes starting with AEAAACEE which are loaded into the look-ahead window.

Search for a sequence in the sliding window that begins with the byte in look-ahead position 0 ("A"). Such a sequence of five bytes starts at sliding window index 6 ("AEAAA")

These five bytes can be replaced by an (Offset,Length) record as shown below:

```
ABBAAAEAAAEEFFFAAAG (6,5)
```

The Sliding window is then shifted over five bytes:

```
     Sliding window    | Look-Ahead window
AAEAAAEEFFFAAAGAEAAA   |   CEGHHHHE ← shift in 5 new bytes
           1111111111
01234567890123456789      01234567 ← Window position
```

Five bytes from the look-ahead window are moved into the sliding window, and five new bytes from the input source are shifted into the look-ahead sliding window.

Decompression Example

Decompressing an LZ77 compressed data stream requires the same size fixed-size buffer for a sliding window; a look-ahead window is not required. To decompress the compressed data sequence:

AABBAAAEAAAEEFFFAAAG (6,5)

First shift uncompressed data into the sliding window FIFO:

```
      Sliding window
AABBAAAEAAAEEFFFAAAG    ← (Offset,Length) record = (6,5)
          1111111111
01234567890123456789    ← Window position
```

When an (offset,length) record is detected, the decompressor points to the position specified by the offset, and begins copying the specified number of symbols/bytes and shifts these into the same sliding window FIFO. Bytes that shift out of the FIFO are the uncompressed data bytes.

Step 1:

```
      Sliding window
AABBAAAEAAAEEFFFAAAG    ← next data is (Offset,Length) record (6,5)
          1111111111
01234567890123456789    ← Window position
```

Step 2:

```
      Sliding window
A ABBAAAEAAAEEFFFAAAGA  ← Copy/shift byte 1 (A) into FIFO
          1111111111
01234567890123456789    ← Window position
```

Step 3:

```
      Sliding window
AA BBAAAEAAAEEFFFAAAGAE  ← Copy/shift byte 2 (E) into FIFO
          1111111111
01234567890123456789    ← Window position
```

Step 4:

```
      Sliding window
AAB BAAAEAAEEFFFAAAGAEA  ← Copy/shift byte 3 (A) into FIFO
          1111111111
01234567890123456789    ← Window position
```

Step 5:

```
      Sliding window
AABB AAAEAAEEFFFAAAGAEAA  ← Copy/shift byte 4 (A) into FIFO
          1111111111
01234567890123456789    ← Window position
```

Step 6:

```
      Sliding window
AABBA AAEAAAEEFFFAAAGAEAAA  ← Copy/shift byte 5 (A) into FIFO
          1111111111
01234567890123456789    ← Window position
```

Uncompressed data

LZ77 belongs to a class of lossless compression methods wherein the compression and decompression cycle results in an exact replica of the original.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/25/01	1.0	Initial Xilinx release.