XMM Optical Monitor

DEPARTMENT OF PHYSICS
UNIVERSITY OF CALIFORNIA, SANTA BARBARA

Author: Cheng Ho (LANL)

XMM-OM DPU Data Decompression Algorithm

DOCUMENT: XMM-OM/UCSB/TC/0041 25-June-98

Contents

An Un-catalogued Document Provided to XMM SOC and ESOC for Ground Software Contractor Baselining

1 General Overview

The XMM-OM DPU employs an onboard data compression algorithm developed at Penn State in 1990. It was later realized that the algorithm is related to an existing algorithm called Variable Block Word Length (VBWL).

In the VBWL algorithm, a big data set is divided into blocks (typically containing a fixed number of data). For each block, a prediction algorithm is applied to decorrelated the data and the residual (decorrelated) datum is expressed in terms of the maximum number of bits necessary within the block. Since each block is separately formulated, the number of bits per datum and the compressed block length will be variable. This is where the name Variable Block Word Length is derived. In a typical implementation, one would select a prediction scheme which is applied throughout the data set.

The independently developed XMM-OM DPU compression algorithm enhances the compression efficiency by 1) dynamically selecting the prediction algorithm amongst three possible schemes for each block and 2) employing a tiered word length structure (hence the name Variable Block Tiered Word Length compression algorithm). The fundamentals of the VBTWL algorithm are described in the XMM document "DPU Processing for XMM/OM: Tracking and Compression Algorithm," XMM-OM/PENN/TC/0004.02). In simple terms, the VBTWL compression process consists of the following steps: determination of the optimal prediction scheme and tiering structure, application of the prediction scheme to decorrelate the data, and encoding of the decorrelated data according to the tiering structure. In this document, we describe the data format as generated by the already implemented on-board compression algorithm, which leads directly to the reversal of the compression process, i.e. the decompression implementation on the ground.

The XMMOM DPU onboard compression package breaks a data set into separate blocks. Each blocks contains up to 256 values, which could have either 24-bit or 16-bit dynamic range. Without loss of generality, we shall refer to each unit of data as pixels. The output of the compression algorithm is a compressed block which consists of a fixed size 128 bit long VBTWL header and a compressed bit stream. The VBTWL header contains all of the information necessary for the decompression. In section 2, we describe the structure of the VBTWL header. In section 3, we describe the decoding and reconstitution processing of the compressed bit stream.

Before we discuss detailed VBTWL decompression, we should stress the following very important points.

1. Compressed Data Word: The XMMOM DPU implementation of VBTWL uses 16-bit words for compressed data storage, which is also the base word length of communication between DPU and ICU. Thus the compressed VBTWL bit stream for each compressed block is padded to the nearest 16-bit word boundary.

2. Endean: As the DPU uses the Motorola 56001 DSP, the endean convention follows the Motorola scheme: Most significant bit as well as most significant byte are transmitted first. To date, we have found that the endean scheme is compatible between the DPU generated data acquired through a Motorola 68030 based EGSE and analysis code developed on a SPARC platform. This endean scheme is however, incompatible with an Intel x86 platform running Linux. Special care must be taken in implementing portable codes across platforms.

Table 1: VBTWL Header Fields

Field Name	Length (bits)	Offset (bits)
block_id	16	0
n_bits	16	16
n_tiers	3	32
im	3	35
blocksize	10	38
it1	5	48
it2	5	53
it3	6	58
nlx2	8	64
nlx3	8	72
vbtwl_sync	16	80
predictor	32	96

3. Decompressed Data Word: The intrinsic word length of XMM-OM DPU is 24-bit. For most modern processors, the appropriate representation that can encompass 24-bit words is the standard 32 bit long integer. This documentation assumes that the decompression implementation is targeted to yield data in a 32-bit integer format.

4. Observation Information: Adhering to the philosophy of software modularity, the compressed data blocks contain NO information related to exposure and operation, other than the sequential block id within a data set. Instead, information such as the OM exposure ID, the science data type, and data set id will be contained in a separate and standard header attached in front of each data set. This separate data set header will NOT be compressed. This document assumes that, as an input to the decompression algorithm, the many layers of headers/trailers added by the DPU-ICU and ICU-TM interface have been properly removed resulting in a concactnation of VBTWL compressed data blocks.

2 VBTWL Header Structure

2.1 Fields in VBTWL Header

The XMMOM DPU flight VBTWL header is 128 bits long and contains the following fields in Table 1.

The meaning of these fields are:

1. block_id: assigned by the calling routine, meant to be a sequential number. For example, a data set containing 1024 pixels generates 4 VBTWL data blocks with block_id 0 to 3.

2. n_bits: number of bits in the compressed VBTWL block, including the VBTWL header length of 128 bits. There will be redundant information in the VBTWL header for consistency check. See below for more details. As mentioned before, the compressed data is based on 16-bit words, thus, the compressed bit stream will contain [(n_bits - 128 + 16 - 1)/16] 16-bit words.

3. n_tiers: number of tiers in the encoding. n_tiers has a value between 1 and 3.

4. im: prediction mode. Flight implementation of XMMOM DPU VBTWL algorithm supports three prediction modes: minimum (0), mean (1), and previous (2).

5. blocksize: number of pixels in this data block. For XMMOM DPU's implementation of VBTWL, this number will be 256 for all data blocks in a data set, except for the last block in the data set which will be the remainder.

6. it1: number of bits for tier 1.

7. it2: number of bits for tier 2. If it2 equals it1, then there is only one tier.

8. it3: number of bits for tier 3. If it3 equals it2, then there is only one or two tiers.

9. nlx2: number of pixels in tier 2.

10. nlx3: number of pixels in tier 3.

11. synch_vbtwl: a synch word in the header. For XMMOM DPU, this is fixed at the hexadecimal number 0xBEEF.

12. predictor: the predictor value of the block.

2.2 Constraints and Consistency Checks

All of these VBTWL fields are non-negative numbers.

There are a number of checks for internal consistency of the header:

1. The synch word (0xBEEF) is located at a fixed location (offset 80).

2. Under normal circumstances, the block_id will be a monotonically incrementing number starting from 0, and all blocksize should be 256 except at the very end of the data set. Situation of out-of-synch block_id or blocksize less than 256 in the middle of a data set indicates anamoly.

3. For XMMOM DPU, all data have a dynamic range of at most 24 bits. Thus the bits of tiers should follow the relationship: it1 <= it2 <= it3 <= 24.

4. If it2 = it3, then n_tiers <= 2. If it2 = it1, then n_tiers = 1. n_tiers has a value between 1 and 3, inclusive.

5. The sum (nlx2 + nlx3) must be less than blocksize. The number of pixels in tier 1 is (blocksize-nlx2-nlx3) and is NOT explicitly expressed in the VBTWL header.

6. All pixel values in tier 1 are expressed in it1 bits. All pixel values in tier 2 are expressed in (it1 + it2) bits. All pixel values in tier 3 are expressed in (it1 + it3) bits. Thus we will have the following very important equality:

n_bits = 128 + blocksize * it1 + nlx2 * it2 + nlx3 * it3

Among these relationships, the decompression algorithm should at least implement two checks: 2.2.1. synch word, and 2.2.6. the n_bits equality.

3 Decoding and Reconstitution

3.1 Tiering Flags

After we read in the padded-to-16-bit-boundary compressed data stream, two distinct flags need to be constructed, each is a left padded 32-bit representation of an it1-bit number. These are flags used to denote the fact that the subsequent bits should be read as second or third tier numbers. The flag for second tier is

flag2 = 2**it1 - 1,

The flag for the third tier is prediction mode dependent. If the prediction mode is 0 (i.e. minimum), then

flag3 = flag2 - 1.

If the prediction mode is 1 or 2 (i.e. mean or previous), then

flag3 = 1.

This bifurcation is caused by a fundamental difference between prediction modes 1 and 2 and mode 0. The mean and previous prediction mode yield decorrelated data that are signed (i.e. both positive and negative). For robustness, the XMMOM DPU implementation of the VBTWL scheme uses the least significant bit as the sign bit for the signed values. And it turns out that flag3 (being negative zero under this implementation) is a unique and otherwise unused number.

3.2 Decoding Algorithm

The operation of decoding a compressed data bit stream is described in the following "flow chart":

1. Extract it1 bits from the compressed bit stream. Cast the it1 bit word to 32 bits, call it t.

2. If the word t equals flag2, then extract the next it2 bits, cast to 32 bits, call it t.

3. Otherwise, if the word t equals flag3, then extract the next it3 bits, cast to 32 bits, call it t.

4. If the prediction mode is 0 (minimum), then the decorrelated value is t.

5. If the prediction mode is 1 or 2 (mean or previous), then the decorrelated value is calculated as follows:

5.a. if t is even, t = t/2,

5.b. if t is odd, t = -(t/2),

3.3 Reconstitution Algorithm

Based on the decoded decorrelated value t, the original value is reconstituted as follows:

3.3.1. If the prediction mode is 0 or 1 (minimum or mean) , then the original pixel value is (t + predictor).

3.3.2. If the prediction mode is 2 (previous), then the original value of the pixel is (t + value of the previous pixel). The original value of the first pixel is the predictor.