HWA2.0 Compression Addendum

Rev. 1.0

Contents

1. Module Overview

The TI Radar Data Compression/Decompression Module (referred henceforth simply as the 'compression module') consists of

1) a compression engine which takes a fixed number of samples and returns a 'block of bits' such that the block's size (in bits occupied) is a fraction of the size of the input samples, and
2) a decompression engine which when provided with the a compressed block of bits, regenerates the original samples (with a possibility of some quantization error).

The features of the compression module are

1) The compression module is designed to achieve *arbitrary* compression ratio. Note that 'compression ratio' is defined as the ratio of 'average bit-width per sample' after compression and 'the original bit-width' before compression. For e.g., a 33 % compression-ratio, results in the average bitwidth after compression being 1/3rd of the bitwidth before compression.

2) It has one configurable algorithms for compression and decompression.
    - Exponential-Golomb Encoding (EGE).

3) It implements a 'block' based compression scheme - i.e. it takes a fixed number of samples (called a block) and creates a 'compressed block of bits' of fixed size. During the Doppler processing operation, when radar data has to be accessed or written in transpose, having each block as a fixed size simplifies the EDMA programming. The EDMA can simply access a full block (across Doppler) in much the same manner as a single range gate.

4) It is a part of the HWA in the IWR6843 device and TPR as one of the programmable 'paths' in the accelerator (in addition to the FFT, CFAR and 'local-max' paths). It can therefore use existing capabilities/resources of the HWA (input/output formatters, state machine, automatic looping, etc).

Note: Low compression ratios can result in 'high quantization noise'. Designers should select the appropriate 'compression ratio' after confirming that it meets the dynamic range necessary for their application.

2. Algorithms

The following section is a brief introduction for the algorithm used in the compression engine.

2.1. Exponential Golomb Encoder (EGE)

The term 'sparse array' is generally used to mean an array where most of the samples are very small, and a few samples are large. Radar data (after the range FFT) is expected to be sparse (in the range dimension). There is typically a few large samples corresponding to target reflections, the remaining samples are either the noise-floor or clutter or weak reflectors and are comparatively small. In a sparse array, the "average" bit-width (where bit-width is defined as the number of bits up to the most-significant 1) required will be small.

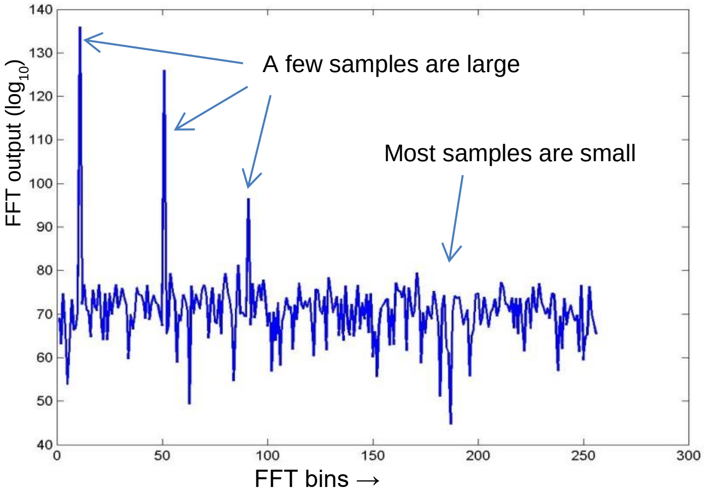
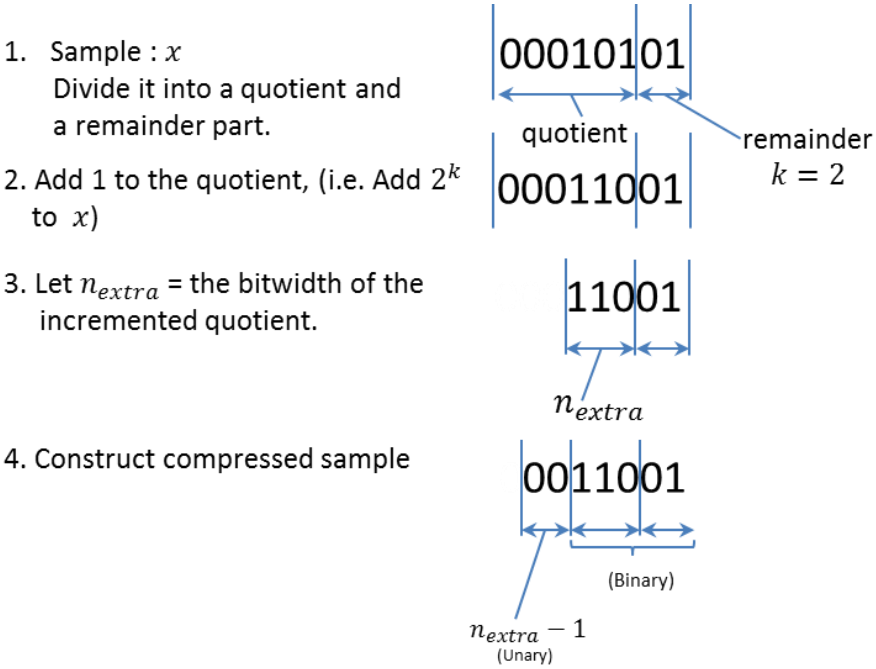


Figure 2 An example of sparse data.

The “Order-k exponential Golomb encoder” (henceforth simply EGE) encodes each sample such that it occupies a space approximately proportional to its bit-width. A description of the algorithm is given in ‘https://en.wikipedia.org/wiki/Exponential-Golomb_coding’. EGE codes are parameterized by the Golomb-parameter ‘k’. This parameter represents the most common bitwidth in the input vector and is required to determine the boundary line between the variable-bitwidth quotient part (that is stored by having its length encoded in unary and the actual bits in binary form) and the fixed-bitwidth remainder part (that is stored in the usual binary form).

For example, if a number, say 23 were to be Exponential-Golomb encoded, then the process would look as follows.



1. Sample : x
   Divide it into a quotient and a remainder part.

2. Add 1 to the quotient, (i.e. Add 2^k to x)

3. Let n_{extra} = the bitwidth of the incremented quotient.

4. Construct compressed sample

Figure 3 Exp-Golomb Encoding Example

One distinction in the compression module is that the order k of the EGE (i.e., Golomb parameter k) is automatically selected from a list of possible values (stored in an array called the 'Golomb parameter array') to optimize the Golomb parameter based on the input samples.

2.1.1. The EGE format

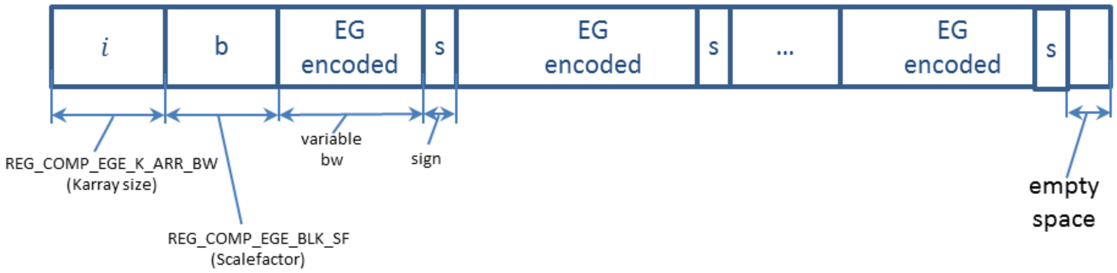


Figure 4 EGE Format

The block starts with a header, holding the scalefactor 'b', and the index to the Golomb parameter array 'i' that is used to encode the block. The rest of the block is filled with EGE words that take a variable number of bit-widths. Since the encoding is done only on the absolute value of each sample (i.e., without sign bit), the sign bit is stored separately after each EG-E word.

3. Operation

The compression/decompression operation is intended to be fairly invisible in operation. In order to use the compression/decompression engine the following steps are necessary.

1) One 'param-set' of the HWA is configured to use the compression path. This configuration includes the number of samples per block, the access pattern, the number of blocks to be compressed, and some additional parameters for the compression engine.
2) Before the compression engine is run, samples (real/complex) are to be placed in an input buffer of the HWA either as part of the operation of a previous param-set by the HWA or by the EDMA. Likewise before the decompression engine is run, compressed blocks are to be placed in an input buffer.
3) The HWA is then executed. When the param-set corresponding to compression is reached the samples in the input buffer are read and compressed and then written to an output buffer as contiguous blocks. When the param-set corresponding to decompression is reached the samples in the output buffer are read, decompressed and then written to the output buffer.

Note : Internally, compression is accomplished via a two-pass operation. In the first pass, the samples are analyzed and the optimal parameters are selected. In the second pass, the samples are compressed to generate the compressed block of bits.

3.1. Configuring compression.

The following steps are to be followed to configure Compression

1) In order to configure the 'compression module', set the compression path in the HWA by setting the ACCEL_MODE to 2, and then set the register CMP_DCMP to 0 to select the compression engine.

2) Enable dither (CMP_DITHER_ENABLE = 1), enable both first and second pass (CMP_PASS_SEL = 3), and enable the header (CMP_HEADER_EN = 1).
3) Configure the 'scale factor bitwidth'. For most cases, it should be set to the logarithm (in base 2) of number of bits per real sample. I.e. for e.g. CMP_SCALEFAC_BW = 4, if the sample bitwidth is 16bits (because $2^4 = 16$), and CMP_SCALEFAC_BW = 5, if the sample bitwidth is 32 bits.
4) Configure BCNT to be 'number of blocks to be compressed'-1. For example, if there are 256 samples and the number of samples per block is 2, there would be 128 blocks to be compressed. The BCNT register would then be configured to 127.
5) Setup the input formatter. In particular registers like SRCACNT, SRC_REAL, SRC_16b32b, and SRC_AINDX. SRCACNT should be set to the 'number of samples per block' – 1 (The '-1' in the previous equation comes from the fact that SRCACNT is zero based). SRC_AINDX would correspond to address increment after SRCACNT samples. For example, for compression of 4 complex 16 bit samples, the following configuration should be used.

   SRCACNT = 3 (4 samples)
   SRC_REAL=0 (complex data)
   SRC_16b23b = 0 (16-bit samples)

6) Setup the output formatter. In particular registers like DSTACNT, DST_REAL, DST_16b32b, and DST_AINDX. DSTACNT should be set to the 'compressed data size (in samples)' – 1. (The '-1' in the previous equation comes from the fact that SRCACNT is zero based). For example, if a 50 % compression is required, DSTACNT can be set to ½ (SRCACNT+1)-1. To compress the previous example by 50 %, the following configuration should be used.

   DSTACNT = 1 (2 compressed samples)
   DST_REAL=1 (real data)
   DST_16b23b = 1 (32-bit samples)

7) Select/Configure the compression method. As of now there is one compression algorithm (EGE).

   a. To configure EGE,
      i. Set the compression method (CMP_METHOD) to 0.
      ii. Program CMP_EGE_K_ARR_LEN which holds the length (in log2) of the list of golomb parameters and also CMP_EGE_K_ARR_<n> which holds the actual parameters.

      Point to note: The golomb parameter should correspond to the most common bit-width in the input array. Since radar data has a wide dynamic range, we typically set the golomb parameter list (for 16 bit numbers) to [0 2 4 6 8 10 12 15]. In the worst-case, the most common bitwidth can be as large as bitwidth of the input rhence the 15 at the end. In the case of 32-bit input, we set the list to [0 4 8 12 16 20 24 31]. CMP_EGE_K_ARR_LEN is normally set 3 (the list length is 8).

Points to note:

i. Configuring the input and output formatters: The sample size for the input formatter is dependent on SRC_REAL, and SRC_16b32b. If SRC_16b32b is set to 1 (i.e. 32 bits), then the per-real-sample bit-width is assumed to be 32bit.

   The sample size of the output formatter is likewise dependent on DST_REAL and DST_16b32b. For example if DST_REAL=1 and DST_16b32b=1, then in each clock cycle the compression engine will write 32-bits to the output. If DST_REAL=0, and DST_16b32b=1, then in each cycle (subject to data availability) the compression module will write 64 bits to the output buffer.

   The reason to care about the sample size in this case is that it directly limits the granularity of compression ratio. For example, consider the following scenario, assume 62.5 % compression is desired. In other words if the input is 128 bits (in total), the compressed output should be 80 bits (which is only divisible by 16 – and not by 32 or 64). So the output per-sample bit-width has to be configured to 16 which needs the destination to be configured to DST_REAL = 1, and DST_16b32b = 0.

ii. Computing the actual compression Ratio.

```
bits_per_src_sample = 16
if SRC_16b32b == 1
   bits_per_src_sample = 32
end

if SRC_REAL == 0
   bits_per_src_sample = 2* bits_per_src_sample
end

Input_Block_Size = bits_per_src_sample * (SRC_ACNT + 1)

bits_per_dst_sample = 16
if DST_16b32b == 1
   bits_per_dst_sample = 32
end

if DST_REAL == 0
   bits_per_dst_sample = 2* bits_per_dst_sample
end

Output_Block_Size = bits_per_dst_sample * (DST_ACNT + 1)
```

The compression ratio is then `Output_Block_Size/Input_Block_Size.`

3.2. Configuring decompression.

Decompression is configured almost exactly the same was as compression, except for the following differences.

1. The only difference is that the register CMP_DCMP should be set to 1 to select the decompression engine, and
2. The 'input formatter configuration' and the 'output format configuration' used in 'the compression stage' should be interchanged. For instance,
   a. the SRCACNT used in compression would become the DSTACNT (and vice versa).
   b. SRC_AINDX should be configured so as to jump to the next compressed block, whereas DST_AINDX should be configured to be the size of a sample.

The following steps are to be followed to configure Decompression

1) set the compression path in the HWA by setting the ACCEL_MODE to 2, and then set the register CMP_DCMP to 1 to select the decompression engine.
2) Enable dither (CMP_DITHER_ENABLE = 1), enable both first and second pass (CMP_PASS_SEL = 3), and enable the header (CMP_HEADER_EN = 1).

3) Configure the ‘scale factor bitwidth’. For most cases, it should be set to the logarithm (in base 2) of number of bits per real sample. I.e. for e.g. CMP_SCALEFAC_BW = 4, if the sample bitwidth is 16bits (because $2^4 = 16$), and CMP_SCALEFAC_BW = 5, if the sample bitwidth is 32 bits.
4) Configure BCNT to be ‘number of blocks to be decompressed’-1. For example, if there are 128 blocks to be decompressed, the BCNT register would be configured to 127.
5) Setup the input formatter. In particular registers like SRC_ACNT, SRC_REAL, SRC_16b32b, and SRC_AINDX. SRC_ACNT should be set to the ‘compressed data size (in samples)’ – 1 (The ‘-1’ in the previous equation comes from the fact that SRC_ACNT is zero based). SRC_AINDX would correspond to address increment after SRC_ACNT samples. For example, for decompression of a block of size 64 bits, the following configuration would work.

    SRC_ACNT = 1 (2 samples)
    SRC_REAL = 0 (complex data)
    SRC_16b23b = 0 (16-bit samples)

6) Setup the output formatter. In particular registers like DST_ACNT, DST_REAL, DST_16b32b, and DST_AINDX. DST_ACNT should be set to the ‘number of samples per block’ – 1. (The ‘-1’ in the previous equation comes from the fact that DST_ACNT is zero based). For example, if 50 % compression is required, DST_ACNT can be set to ½ (SRCACNT+1)-1. If there are 4 complex samples in the block, then the following configuration should be used.

    DST_ACNT = 3 (4 compressed samples)
    DST_REAL=0 (real data)
    DST_16b23b = 0 (32-bit samples)

7) Select/Configure the compression method. Note that the configuration for decompression should be exactly the same as the configuration for compression.

    a. To configure EGE,
        i. Set the compression method (CMP_METHOD) to 0.
        ii. Program CMP_EGE_K_ARR_LEN which holds the length (in log2) of the list of golomb parameters and also CMP_EGE_K_ARR_<n> which holds the actual parameters.
        Point to note: The golomb parameter should correspond to the most common bit-width in the input array. Since radar data has a wide dynamic range, we typically set the golomb parameter list (for 16 bit numbers) to [0 2 4 6 8 10 12 15]. In the worst-case, the most common bitwidth can be as large as bitwidth of the input rhence the 15 at the end. In the case of 32-bit input, we set the list to [0 4 8 12 16 20 24 31]. CMP_EGE_K_ARR_LEN is normally set 3 (the list length is 8).

3.3. Speed

For EGE, Compression of one complex sample takes one cycle, along with a one-time latency of ‘number of samples in a block’ for every param-Set that is run. At the end of each block, 4 extra cycles are taken to flush out the remaining samples in IWR6843 (In TPR these extra cycles do not exist). Again, the number of cycles can be computed as ~(BCNT+2)(max(SRC_ACNT,DST_ACNT) + 1)

3.4. Register Descriptions

1) Basic configuration.

Register	Width	Parameter-Set? (Y/N)	Description

<table>
<tr><td>ACCEL_MODE</td><td>3</td><td>Y</td><td>The accelerator core is essentially a parallel set of paths. Each path performs a certain core operation, either FFT, CFAR-CA, compression/decompression, etc. To select the Compression/Decompression Module set this register to 2.</td></tr>
<tr><td>CMP_DCMP</td><td>1</td><td>Y</td><td>This register controls the compression mode, i.e. whether the operation to be performed is compression (when CMP_DCMP = 0) or decompression (when CMP_DCMP = 1).</td></tr>
<tr><td>CMP_METHOD</td><td>3</td><td>Y</td><td>3 bit register that selects one of the two compression algorithms. The only valid value of the register is:<table><tr><td>Value</td><td>Description</td></tr><tr><td>0</td><td>EGE</td></tr></table>All other values are to be considered invalid</td></tr>
<tr><td>CMP_ERR</td><td>1</td><td>N</td><td>A read-only register that is set whenever the compression procedure encounters an irrecoverable error. This can happen in the case of EGE, when the golomb parameters are incorrectly coded. A separate CLR_CMP_ERR register bit is provided, so that the error flag can be cleared. Note that an error can happen if an invalid configuration is used, such as an extreme configuration of the compression engine (for example, a very low impractical compression ratio is configured). (Only available in TPR and only for EGE).</td></tr>
<tr><td>CLR_CMP_ERR</td><td>1</td><td>N</td><td>Clear the CMP_ERR error flag. See description of CMP_ERR. (only in TPR)</td></tr>
</table>

2) Compression/Decompression

<table>
<tr><th>Register</th><th>Width</th><th>Parameter-Set? (Y/N)</th><th>Description</th></tr>
<tr><td>CMP_DITHER_ENABLE</td><td>1</td><td>Y</td><td>The register enables dithering. Dithering prevents periodic quantization patterns from resulting in spurs. The dither source provides 3 bits of dither for every sample.
Note : This register has to be set to 1 for proper operation of the hardware</td></tr>
<tr><td>CMP_PASS_SEL</td><td>2</td><td>Y</td><td>This register optionally bypasses the first pass (i.e. optimization of parameters) or the 2nd pass (actual compression). Note that if the first pass is bypassed, the programmed scale-factor is used.<table><tr><td>Value</td><td>Description</td></tr><tr><td>11</td><td>Both first pass and second pass are enabled.</td></tr><tr><td>01</td><td>First pass is disabled.</td></tr></table></td></tr>
<tr><td>CMP_HEADER_EN</td><td>1</td><td>Y</td><td>Optionally populate the header in the compressed stream. In most normal use-cases, this is set to 1, as the header is necessary for decompressing a block. However, if the first-pass is bypassed and all blocks are configured to use a specific customer-chosen scale-factor value, the header wouldn't be necessary.</td></tr>
<tr><td>CMP_SCALEFAC_BW</td><td>4</td><td>Y</td><td>The number of bits to be used in the header for the 'common scale-factor' per block. If the input is 16-bit (real or complex) set the common scale-factor to 4 (since the scale factor can vary from 0 to 15). If the input is 32-bit (real or complex), set the complex scale-factor to 5 (since the scale factor can vary from 0 to 31).</td></tr>
</table>

3) Some of the Registers from the input and output formatters are reused internally to compute the compression ratio (with some additional description).

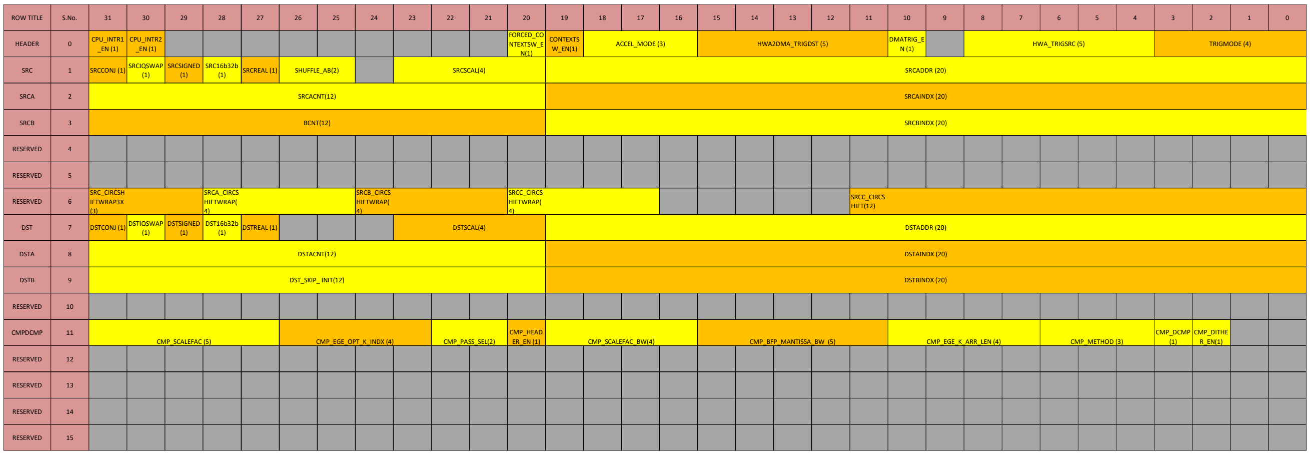
Register	Width	Parameter-Set? (Y/N)	Description

SRC_ACNT	12	Y	This register (plus 1) denotes the number of samples in a block. This is a zero-based count and therefore a register value of 15 indicates that there are 16 samples in a block. Note that we also rely on SRC_REAL and SRC_16b32b to denote the size of each sample, and they have to be correctly programmed for SRCACNT to select the necessary samples. Also, the maximum number of samples has to fit in the input buffer of the compression engine (< 2Kb).
DST_ACNT	12	Y	This register (plus 1) denotes the desired output size in samples. To get the true compressed size in bits, DST_REAL, and DST_16b32b should be taken into consideration. When using the EG algorithm, the compression engine will compress all data so that it fits within this DSTACNT.
BCNT	12	Y	This register (plus 1) denotes the number of blocks in the input buffer (ping and pong).

4) EGE specific registers.

Register	Width	Parameter-Set? (Y/N)	Description
CMP_EGE_K_ARR_LEN	4	Y	This register value encodes the length of the 'list of Golomb parameters' to optimize over. The actual length of this list is 2^(Register Value). The valid range for this register is from 1 to 3. In effect the valid length of the list of parameters is 2, 4, 8.
CMP_EGE_K_ARR_<n>	5 bits per element (upto 8 elements in total)	N	These set of registers hold the list of golomb parameters to optimize over. The number of valid elements is determined by the CMP_EGE_K_ARR_LEN (see register).

3.4.1. parameter sets



ROW TITLE	S.No.	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HEADER	0	CPU_INTR1_EN (1)	CPU_INTR2_EN (1)										FORCED_CONTEXTSW_EN(1)	CONTEXTSW_EN(1)	ACCEL_MODE (3)			HWA2DMA_TRIGDST (5)					DMATRIG_EN (1)		HWA_TRIGSRC (5)					TRIGMODE (4)			
SRC	1	SRCCONJ (1)	SRCIQSWAP (1)	SRCSIGNED (1)	SRC16b32b (1)	SRCREAL (1)	SHUFFLE_AB(2)			SRCSCAL(4)				SRCADDR (20)																			
SRCA	2	SRCACNT(12)												SRCAINDX (20)																			
SRCB	3	BCNT(12)												SRCBINDX (20)																			
RESERVED	4																																
RESERVED	5																																
RESERVED	6	SRC_CIRCSHIFTWRAP3X (3)			SRCA_CIRCSHIFTWRAP(4)				SRCB_CIRCSHIFTWRAP(4)				SRCC_CIRCSHIFTWRAP(4)									SRCC_CIRCSHIFT(12)											
DST	7	DSTCONJ (1)	DSTIQSWAP (1)	DSTSIGNED (1)	DST16b32b (1)	DSTREAL (1)				DSTSCAL(4)				DSTADDR (20)																			
DSTA	8	DSTACNT(12)												DSTAINDX (20)																			
DSTB	9	DST_SKIP_ INIT(12)												DSTBINDX (20)																			
RESERVED	10																																
CMPDCMP	11	CMP_SCALEFAC (5)					CMP_EGE_OPT_K_INDX (4)				CMP_PASS_SEL(2)		CMP_HEADER_EN (1)	CMP_SCALEFAC_BW(4)				CMP_BFP_MANTISSA_BW (5)					CMP_EGE_K_ARR_LEN (4)				CMP_METHOD (3)			CMP_DCMP (1)	CMP_DITHER_EN(1)		
RESERVED	12																																
RESERVED	13																																
RESERVED	14																																
RESERVED	15																																

Figure 5 Compression module - parameter-set arrangement (TPR)