Application Report
SWRA663–December 2019

Memory Compression and Decompression Engine for TI mmWave Radar

Anil Mani and Zigang Yang

ABSTRACT

This white paper introduces the notion of compressing radar data to improve performance, where performance can mean anything from improved maximum range to better velocity resolution, without any compromise on other parameters of interest. This performance improvement is achieved as a consequence of the increased amount of radar-data that can be stored on the device, which would otherwise require larger on device RAM. Using examples and theory, this paper shows that for many common scenarios, compression is a safe and effective way to virtually increase the amount of data that can be stored in the onboard RAM.

This paper also introduces the TI Memory Compression/Decompression Engine, a hardware module that is newly added to the TI millimeter wave radar solutions. It is included in the IWR6843 and other upcoming radar devices that contains hardware accelerator (HWA) version 1.1 or above. The Hardware Accelerator (HWA) version for each silicon device can be found in Technical Reference Manual.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

FMCW Radar processing (see [1]) requires the storage of a data structure called the Radar Cube (see Figure 1), a three dimensional array with dimensions of range-bins × chirps × Rx antennas. To completely process radar signals, the entire radar cube must be preserved for the majority of the radar processing flow. Therefore, compressing radar data means essentially compressing radar cubes.

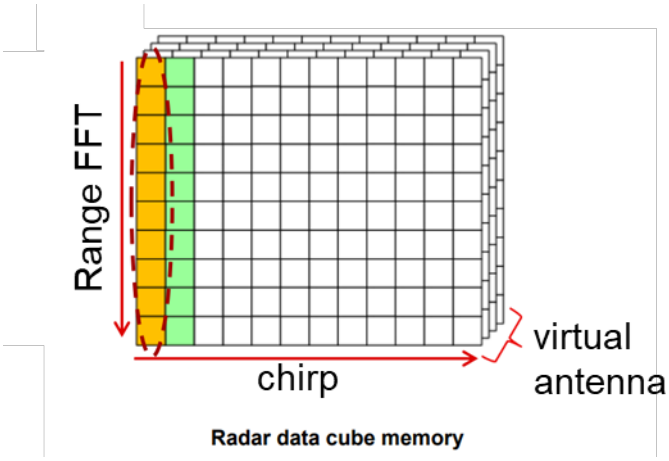

Figure 1. A Typical Radar Cube

The intent of this document is to convey that compression is a safe and effective way to virtually increase the amount of data that can be stored in the onboard RAM. As an example, see Figure 2, where two range-doppler images are shown. The image on the left shows the radar data compressed by 40%. The image on right is uncompressed. Note how even 40% compression maintains the key features of the range-doppler image.

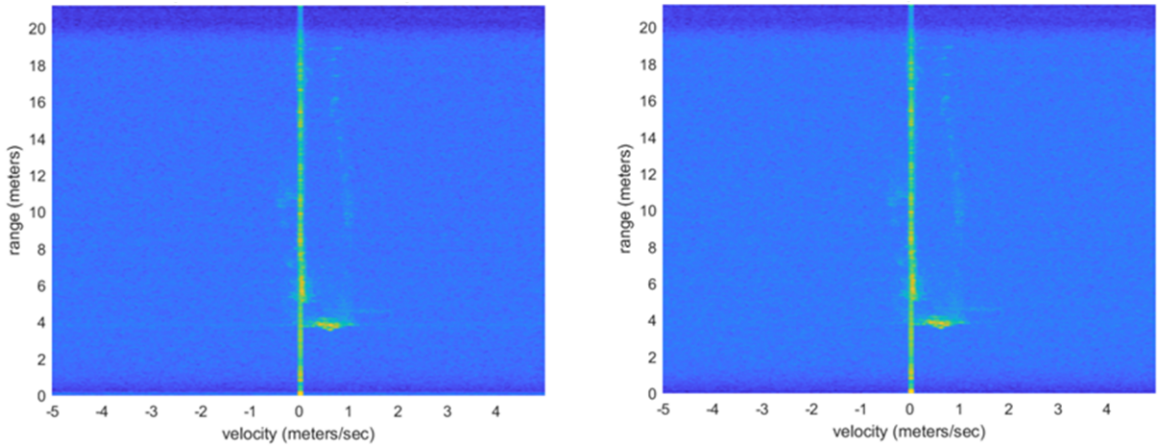

Figure 2. Two Range-Doppler Images

The white paper is organized as follows. Section I describes the need for larger storage of radar-data, and thus compression. This section also covers properties of radar data that make it amenable to compression. In particular, the concept of per-range-bin dynamic range, a key parameter for determining the viability of compression is introduced. Section II introduces TI's radar-data compression/decompression engine which is based on an exponential golomb encoder (or EGE). Sample results of the impact of compression on radar processing results are provided. Also provided is online collateral, which lets anyone interested in using TI's compression engine test it in their processing flows.

2 Compression in the Radar Processing Flow

First, a short review of the initial stages of FMCW Radar processing. As shown in Figure 3, a frame consisting of the ADC data from a number of chirps (such as N_C chirps) is collected and processed in two distinct stages – inter-chirp processing and inter-frame processing.

In inter-chirp processing, the ADC data of each chirp undergoes an FFT (called the range-FFT), and is stored in the device's memory as one portion of the radar-cube (let each range-FFT output's length be N_{RFFT}). If multiple receivers have been enabled, a separate range-FFT must be performed for each receiver's ADC data (assume N_{Rx} receivers) and stored in memory. In such a manner as more chirps are processed the radar-cube is progressively filled up. The dimensions of the radar cube at this stage are range-bin × chirp number × receiver id, and the total size (in samples) of this radar cube once the inter-chirp processing stage is complete is $N_{RFFT} \times N_C \times N_{Rx}$. In xWR devices, the memory where the radar-cube is stored is typically the L3 cache.

When all chirps have been processed and stored in the radar-cube, the next stage of processing (inter-frame processing) can begin. In this stage, each range-gate (a vector consisting of single range-bin across all-chirps) is retrieved and undergoes an FFT (called the 2nd dimension FFT, or Doppler-FFT).

To access a range-gate, the order of the radar-cube's dimensions must be changed (from range-bin × chirp number × receiver id to chirp number × range-bin × receiver id). This is called a transpose operation, and can be either performed during inter chirp processing (as part of writing to memory), or during inter-frame processing (as part of reading from the radar-cube).

Thus, as shown in Figure 3, compression should be performed on the range-FFT output, prior to its storage in memory. The compression algorithm must work on a per-chirp basis on the range-FFT output of the data from all receivers. The decompression algorithm must read the compressed radar cube in transpose and deliver the range-gates for Doppler processing.

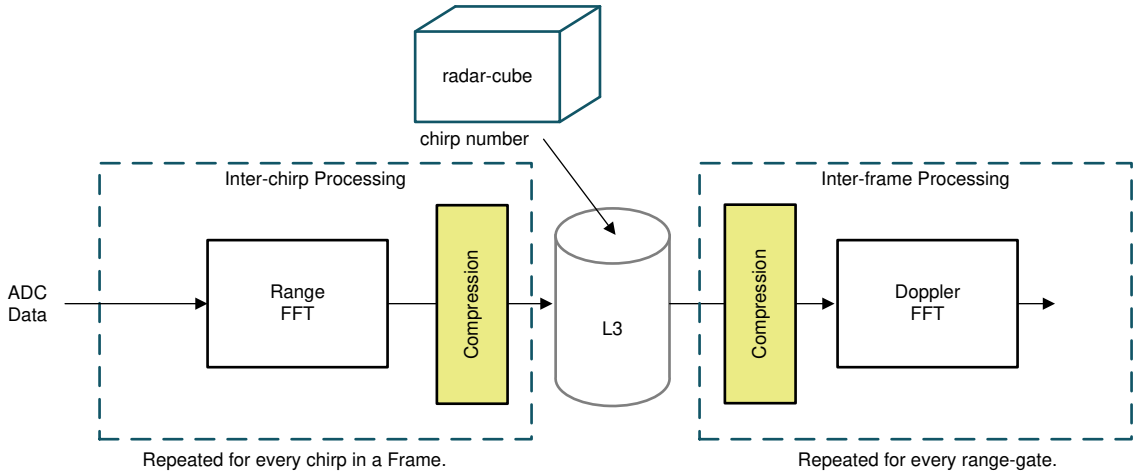

Figure 3. Standard Radar Processing Flow

2.1 Motivation for Larger Radar Cube

In simple terms, a larger radar-cube could enable at least one of the following improvements:

- Increase N_c (the number of chirps), which results in better velocity resolution (δv) for a given max-velocity ($V_{max}M$), or better max-velocity for a given velocity resolution. This can be seen from the equation relating max-velocity and velocity resolution: $\delta v = V_{max} / (2N_c)$.
- Increase N_{RFFT} (the range-FFT size), which results in finer (smaller) range resolution (δR) (assuming that the RF bandwidth ($B_{max}M$) of the device supports it) for a given max-range ($R_{max}M$) or vice-versa. This can be seen from the formula relating δR and R_{max} : $\delta R = R_{max}/N_{RFFT}$.
- Increase N_{rx}, the number of Rx antennas which results in better angular resolution (assuming that the physical device has more receivers/antennas) without affecting N_{RFFT} and N_c.

2.2 Considerations When Applying Compression

2.2.1 The Need for Efficient Transpose Access (Block-Based Schemes)

As described in Section 2, Doppler processing requires the extraction of a single range-gate, through a transpose access operation. Therefore, it is not practical to compress an entire range-FFT to a single unit, and then in the intra-frame period decompress it to extract one sample, and then repeat the process for every chirp to get a range-gate.

Instead, samples (across *adjacent* range-bins or rx-antennas or chirps) are grouped together into blocks. Each block is then compressed to a fixed size. Thus, the transpose access of blocks across a range-gate can be done simply and cheaply. Because each block has a small number of samples (say N_B), the memory requirement when an entire range-gate is decompressed is at most N_B N_c.

After forming a block, a lossless compression operation (described in Section 3) is performed. If even after compression, the compressed output does not meet the desired compression ratio, a minimum number of LSBs are dropped from each sample to meet the compression ratio and ensure that each compressed block is of a fixed size. The number of LSBs dropped are also noted in the compressed block because it is needed for decompression.

2.2.2 Compressing Across Antenna and Across Range

The block of samples to be compressed can be selected from any one of the three dimensions available (range, chirp number, or rx antenna) or a combination of two of these dimensions. There are different advantages and disadvantages to selecting the dimension to be used.

- **Compression across chirps**: As described in Section 2, inter-chirp processing consists of performing a range-FFT for each of the RX antennas. The range-FFT samples for each range-bin across the different RX antennas are compressed into a single block. This is the simplest way of incorporating compression into the signal processing chain. The compression operation uses only data available at a single inter-chirp processing epoch. Likewise decompression is simple because each compressed block consists of data for a single range-gate which can be directly consumed for the computation of the Doppler-FFT.
- **Compression across chirps**: In addition to samples across RX antennas, each compressed block, can also include samples for the same range-bin across two or more adjacent chirps. This introduces some buffering requirement during inter-chirp processing (to store range-FFT data across multiple inter-chirp epochs). However, since each compressed block still contains data for a single range-gate, the complexity during de-compression remains unchanged.
- **Compression across range**: In many applications range-FFT data can be sparse (see for example, Figure 3) . Allowing the compressed block to include data across multiple range-bins enables the compression algorithm to benefit from this sparsity. Since each compressed block now contains data for multiple range-gates, this introduces a buffering requirement during the decompression.

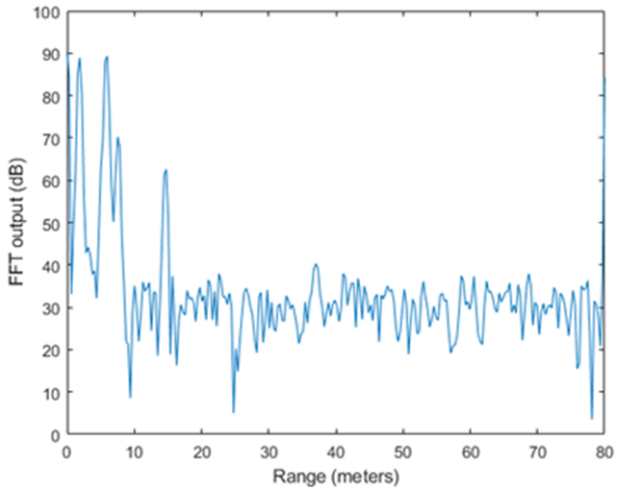

Figure 4. Typical Range FFT Output

2.2.3 Lower Dynamic Requirements per Range-gate

Compression works because in most use-cases dynamic range per range-bin is limited.

Dynamic range is the ratio of the largest sample in a vector to the noise-floor. It can be calculated using the per-sample bitwidth (N_{bits}) of the vector as N_{bits} × 6.0 dB. If N_{bits} is 16-bits, the dynamic range possible is ~96 dB. Typical range-FFT data has a high dynamic range (90 dB or more), because it can receive strong reflections from nearby objects as well as weaker reflections from objects at far distance.

However, within a range-bin, the RCS (radar cross section) variation of valid targets is small (30 dB or less). It can be derived by considering RCS of the strongest target (a truck at 25 dBsm) versus the weakest target (a person at –5 dBsm). To detect the weakest object the SNR of that target must be reasonable (~ 15 dB). Thus, the necessary dynamic range requirement for a range-bin is the RCS variation + the SNR required and is of the order of ~45 dB (which requires only 8 bits per-sample - compared to about 16 bits in the previous case).

To take advantage of the fact that dynamic range requirements are reduced for a single range-bin, samples that are close-by in range (preferably in the same range-bin across different Rx channels) are grouped together to create a block.

Because EGE is a variable bit-width compression scheme, it is difficult to compute the dynamic range after compression. However, the user can easily compute limits. For example, if a block of 4 16-bit numbers is compressed to half its size, then bitwidth per compressed sample would be 8-bits. Because some of that space is used for headers, in practice the bitwidth would be most-likely 7 or 6 bits per sample – leading to either 42 dB or 36 dB of dynamic range.

In this case, the requirement of 45 dB of per-bin dynamic range is not met. However, At this point in the processing flow, only the range FFT has been performed. The dynamic range further improves with the Doppler processing (by $10log_{10}(N_c)$). If N_c = 256 (i.e $10log_{10}(N_c)$ = 24) then the dynamic range would improve from 36 dB to 60 dB (assuming the worst-case of 6 bits per compressed sample).

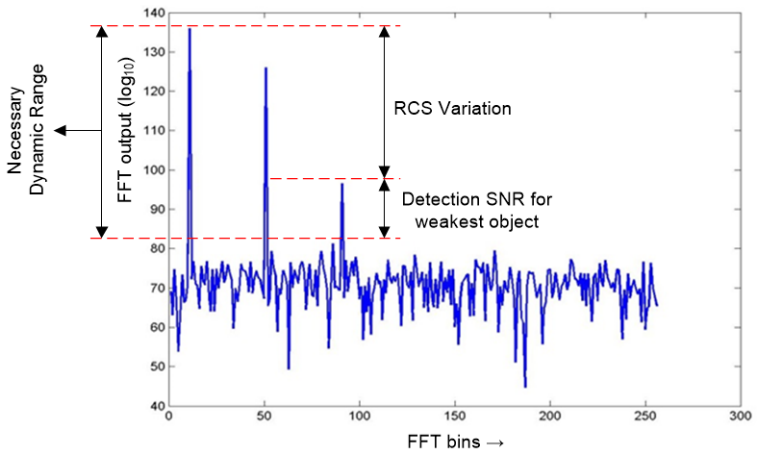

Figure 5. Dynamic Range Necessary for Proper Radar Operation

3 The TI Compression Engine

3.1 Features

The TI compression engine on the IWR6843 device is with following features:

- The compression engine is designed to achieve arbitrary compression ratio. Compression ratio is defined as the ratio of average bit-width per sample after compression and the original bit-width before compression. That is, a 33 % compression-ratio results in the average bit-width after compression being 1/3rd of the bit-width before compression. In other words, lower compression ratios save more memory, but result in more quantization loss.
- It uses Exponential Golomb Encoding (EGE), which helps exploit sparsity in the range dimension and an optimization step which selects the best Golomb parameter based on the most common bit-width. It works with both spiky data and nearly constant data. It can compress and decompress one complex sample per cycle.
- It implements a block-based compression scheme: that is, 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 must be accessed or written in transpose, having each block as a fixed size simplifies data access. The DMA can access a full block (across Doppler) in the same manner as it accesses a single range gate.
- It is a part of the radar hardware accelerator (or HWA) version 1.1 as one of the programmable paths in the accelerator (in addition to the FFT, CFAR, and local-max paths). It can thus use existing capabilities and resources of the HWA (input/output formatters, state machine, automatic looping, and so forth).

3.2 Exponential Golomb Encoding (EGE)

The compression algorithm used in the engine is the Order-k exponential Golomb encoder, which encodes each sample such that it occupies a space approximately proportional to its bit-width. A description of the algorithm is given in [2]. Order-k Exponential Golomb 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 such as 23 were to be Exponential-Golomb encoded (using k = 2), the process would look as shown in Figure 6.

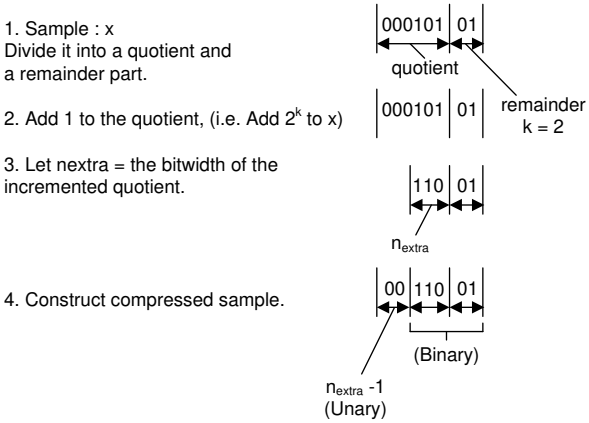

Figure 6. Exp-Golomb Encoding Example

In the compression engine the Golomb parameter k is optimally selected from a user-programmable list. This is done to achieve the most lossless compression possible based on the distribution of the input samples without user intervention during compression.

3.3 Test Results on Field Data

This section presents representative results from field tests. In these results, we show range Doppler images and also cross sections of range-gates where the effects of compression are more easily seen. This section is divided into two parts. The first part provides results where compression was performed in the range dimension, and the second part provides results where compression was performed in the doppler dimension.

3.3.1 Compression in the Range Dimension

In this test, 8 range-bins were grouped into a block and compressed together. EGE performs as expected, using the sparsity of the range dimension to compress radar data, and compression ratios as low as 40% (see Figure 2) are nearly indistinguishable from the no compression radar image. To show a contrast, we show a picture (Figure 7) of a 33% compressed radar cube. Ridges are seen across Doppler (marked by a black square), especially where strong objects (in this case, a car) are located.

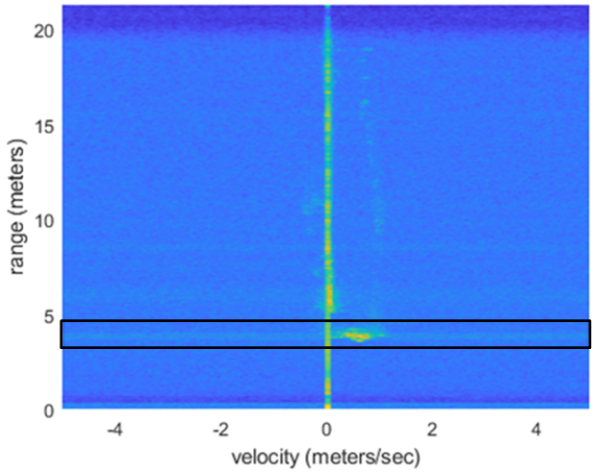

Figure 7. Range-Doppler Image for a 33% Compressed Radar Cube

We extracted the strongest range-gate (after 2D FFT) for both 40% compression and 33% compression to show its effect (Figure 8). As shown, reducing the compression ratio from 40% to 33% has an impact on the noise-floor. However, the important peaks are virtually identical with the no compression case, for both 40% and 33% compression. 40% compression and no compression are both on top of each other, whereas there is a slight degradation for 33% compression.

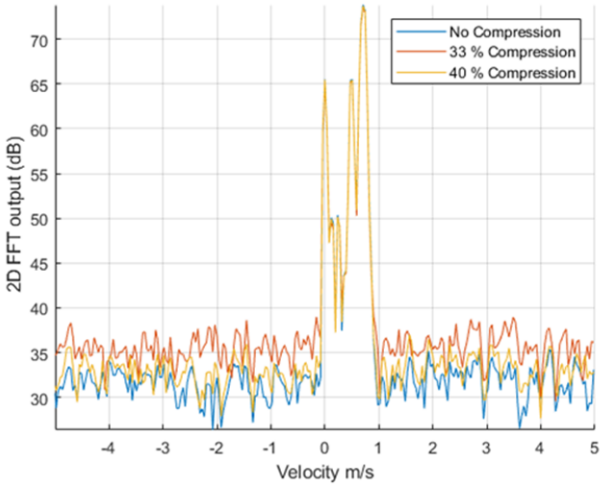

Figure 8. Range Gate Corresponding to the Target of Interest

Over every test, we saw that compression rates of 50% are possible without much degradation. However, compression rates of 33% or more should not affect detection performance if the detection algorithms can ignore these ridges. For example, a two pass CFAR algorithm run both in the range dimension and in the doppler dimension would disallow detections of the ridge.

The dynamic range of this range-gate for the no compression case is ~ 40 dB, which reduces to 37 dB due to the 33% compression.

3.3.2 Compression in the Antenna Dimension

In this test, 4 complex samples (collected across the 4 receivers) for a single range-bin are grouped as block and compressed. Again, for compression ratios of 40%, the performance is fairly good; however, at 33%, we see more dominant ridges (Figure 9) as compared to case where compression was performed in the range dimension.

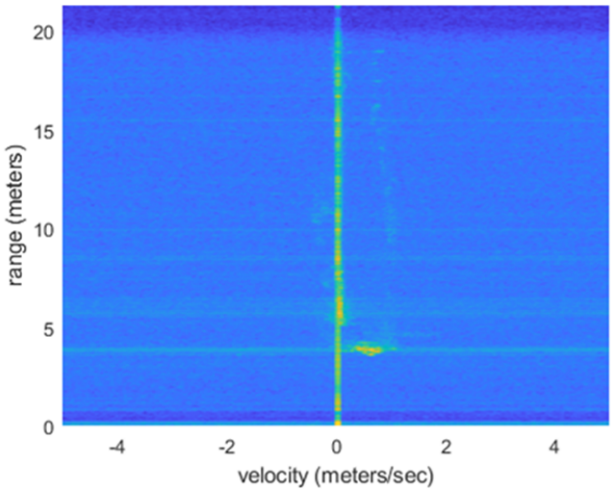

Figure 9. The Range-Doppler Image Resulting From 33% Compression Across Antennas

In the range-gate picture (Figure 10) corresponding to the same target, we see that performance degradation at 33% is worse than the compression across range (Figure 8). In particular, the degradation in the noise floor is higher, reducing the dynamic range to ~35 dB.

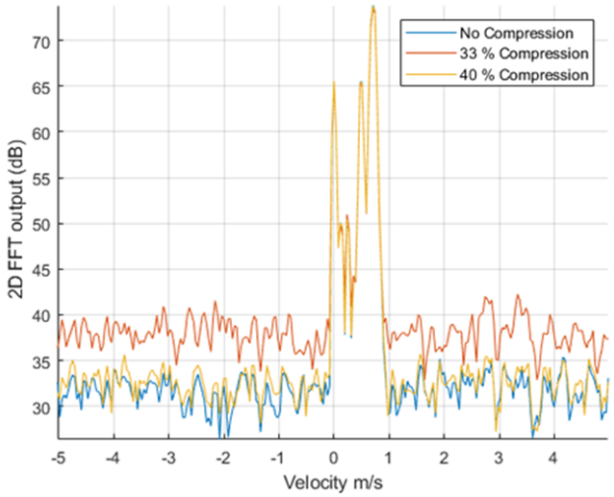

Figure 10. Range-Gate Picture for Compression Across Antennas

In this particular mode, TI recommends using compression rates of 62.5% to compress radar data if no distortion is desired.

4 Conclusion

A compression algorithm can be used to compress radar-cubes and thereby significantly increase the dimensions of the radar-cube that can be stored on a device. Increasing these dimensions can improve important performance parameters (such as maximum range or velocity resolution) and make different applications viable. TI's new compression/decompression engine is part of the radar hardware accelerator version 1.1, starting in the IWR6843 device, can be used to elegantly implement compression and decompression into the radar processing flow. Design considerations based on the desired dynamic range should be taken into account when deciding what compression ratio is acceptable to a particular application.

Collateral in the form of an easily embeddable MATLAB code has been provided (see [3]). Designers can use it to test how compression works in their flows.

5 References

1. *Introduction to FMCW Radar*
2. *Exponential golomb encoding*
3. *Matlab code for the compression engine*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265