**This document includes 2 sections:**

**PART – I:**

**ExafelSZ Example**

(An example on how to run ExafelSZ)

**And**

**PART – II:**

**ExafelSZ Compressor Manual**

(A manual for how to install and run ExafelSZ)

**PART – I:**

**ExafelSZ Example**

**1) Download and prepare the example package**

Download and build SZ. Refer to PART- II: ExafelSZ Compressor Manual for more details. You can use exafelSZ\_example.c from the example package as your program that calls the ExafelSZ functions. Please make sure to define ${SZ\_HOME} environment variable correctly.

Download and untar the example package: exafelSZ\_example.tar.gz

Make all the tools:

make all

See Appendix A for details about the files in the example package.

**2) Run the tools through a single script:**

Set up the exampleRoot variable in exafelSZ\_example.sh.

(It should point to a working folder with enough disk space to hold all the files, including the original data, decompressed data, and intermediate files. The total size needed will be more than 2 times the original data size, but probably much less than 3 times the original data size.)

Examine other variables and commands, and make changes if necessary.

Run exafelSZ\_example.sh:

./exafelSZ\_example.sh

**PART – II:**

**ExafelSZ Compressor Manual**

**1) Download and install SZ**

git clone https://github.com/disheng222/SZ

cd SZ

mkdir install

./configure --prefix=$(pwd)/install

make -j 8

make install

**2) Compilation Instructions**

**2.a) Static Linking:**

Add this to your compilation command:

-I${SZ\_DIR}/install/include ${SZ\_DIR}/install/lib/libSZ.a ${SZ\_DIR}/install/lib/libzstd.a -lz -lm

where ${SZ\_DIR} is the path to the main SZ directory. (See example Makefile)

No need to modify your ~/.bashrc

**2.b) Dynamic Linking:**

Add this to your compilation command:

-I${SZ\_DIR}/install/include -L${SZ\_DIR}/install/lib -lSZ -lz -lzstd -lm

where $(SZ\_DIR) is the path to the main SZ directory. (See example Makefile)

Add this line to your .bashrc:

export LD\_LIBRARY\_PATH=${SZ\_DIR}/install/lib:$LD\_LIBRARY\_PATH

**3) Coding Instructions**

**3.a) Include the header file:**

Include sz.h in your C file:

#include "sz.h"

**3.b) Coding requirements for SZ:**

**i)** Call SZ\_Init(NULL) before your first compress (and decompress)

**ii)** Deallocate compressed buffer when you do not need it anymore

**iii)** Deallocate decompressed buffer when you do not need it anymore

**iv)** Call SZ\_Finalize() after your last compress and decompress

**NOTE:** Compress and decompress functions allocate and return a buffer. The user should deallocate those buffers manually to prevent memory leaks.

**3.b) How to call SZ compress and decompress functions:**

Declare and configure an exafelSZ\_params structure:

exafelSZ\_params pr;

... (see exafelSZ\_example.c for an example)

**NOTE:** You need all the following members of exafelSZ\_params structure to be set up correctly for compression:

binSize, tolerance, szDim, peakSize, calibPanel, peaks

**NOTE:** For decompression, you need every member required for compression except peaks. Peaks array is already saved into compressed file and decompress function reads it from there automatically.

**3.b.i) Compress function convention:**

unsigned char\* SZ\_compress\_customize(const char\* cmprName, void\* userPara, int dataType, void\* data, size\_t r5, size\_t r4, size\_t r3, size\_t r2, size\_t r1, size\_t \*outSize, int \*status)

**Example: (from exafelSZ\_example.c)**

compressedBuffer=SZ\_compress\_customize("ExaFEL",(void\*)(&pr), SZ\_FLOAT,(void\*)origData, 0, nEvents, panels, rows, cols, &compressedSize, &compStatus);

**3.b.ii) Decompress function convention:**

void\* SZ\_decompress\_customize(const char\* cmprName, void\* userPara, int dataType, unsigned char\* bytes, size\_t byteLength, size\_t r5, size\_t r4, size\_t r3, size\_t r2, size\_t r1, int \*status)

**Example: (from exafelSZ\_example.c)**

decompressedBuffer=SZ\_decompress\_customize("ExaFEL",(void\*)(&pr), SZ\_FLOAT, compressedBuffer, compressedSize, 0, nEvents, panels, rows, cols, &decompStatus);

**Appendix:**

**A) Example Package:**

**Makefile:** Includes both static and dynamic linking examples

**exafelSZ\_example.sh:** The script that contains all the basic information to run an example.

**exafelSZ\_example.c:** Demonstrates how to use compress / decompress functions. Also writes compressed and decompressed data to some output files.

**calibPanel.bin:** An empty panel that only contains calibration pixels. Will be used to create the masks. This mask is low-active (0=Active, 1=Not Active).

**bin2cxi.py:** Converts a binary file to cxi file.

**fixPythonBinPeakFile.c:** This tool fixes a problem caused by limited binary file support of Python 2.

**roi\_SZ\_write\_p17.py:** Thisscript reads the data, then generates the data.bin and peaks.bin files.

**extractFirstPanel.c:** This tool extracts the first panel from the given binary data file. This will only be needed if calibPanel.bin file has a problem and the users need to generate it by themselves.

**B) A quick way to confirm the results:**

Run psocake in SZ mode to visualize the cxi file. You should see the ROI regions around the peaks (you can run peak finder to easily identify peaks). ROI values must match the original values exactly, since they are stored lossless.

The background also should look like the binned version of the original data. But it also goes through lossy compression, so the decompressed binned values should be within the specified error bound (15 in our example) of the binned original values.

**C) Possible files of interest:**

**ExafelSZ specific files:**

**${SZ\_DIR}/sz/include/ExafelSZ.h :**

Contains exafelSZ\_params structure definition. Do not include this file! This file is automatically included when you include sz.h

**${SZ\_DIR}/sz/src/ExafelSZ.c :**

Contains the functions definitions . At some point, it calls the original SZ algorithm through SZ\_compress\_args function.

**General SZ suite files:**

**${SZ\_DIR}/sz/include/sz.h :**

This should be included for ExafelSZ to work.

**${SZ\_DIR}/sz/src/sz.c :**

There are only 2 function definitions that can be of interest: **SZ\_compress\_customize**

**SZ\_decompress\_customize**

These are the compress / decompress functions that the user should call.

**D) Explanation of Parameters:**

**NOTE:** There are no default values assigned to parameters. So the user should specify every one of the required parameters appropriately.

Please see the definition of exafelSZ\_params structure from the file $(SZ\_DIR)/sz/src/ExafelSZ.h. The parameters are explained briefly in comments:

typedef struct exafelSZ\_params{

uint8\_t \*peaks;

uint8\_t \*calibPanel;

uint8\_t binSize; //Binning: (pr->binSize x pr->binSize) to (1 x 1)

double tolerance; //SZ pr->tolerance

uint8\_t szDim; //1D/2D/3D compression/decompression

//uint8\_t szBlockSize; //Currently unused

uint8\_t peakSize; //MUST BE ODD AND NOT EVEN! Each peak will have size of: (peakSize x peakSize)

//CALCULATED VARIBALES:

uint64\_t binnedRows;

uint64\_t binnedCols;

uint8\_t peakRadius; //Will be calculated using peakSize

} exafelSZ\_params;

**Dimensions:**

The input data is considered to be 4D: [nEvents] [panels] [rows] [cols], where:

**nEvents:** Number of events per batch. Set to 1 compress / decompress events one by one.

**panels:** Number of panels per event.

**rows:** Number of rows in each panel.

**cols:** Number of columns in each panel.

**NOTE:** In the current version, the user must use fixed dimensions throughout the entire compression/decompression process.

**NOTE:** Even though the user has freedom to set up these variables to almost any value they want, up to so far, we have always used the following dimensions:

panels = 32, rows = 185, cols = 388

**peaks array:**

For each event:

Number of peaks: 1 x uint64\_t

For each peak:

Panel coordinate: 1 x uint16\_t

Row coordinate: 1 x uint16\_t

Column coordinate: 1 x uint16\_t

**calibPanel array:**

Consist of 185x388 uint8\_t elements.

Provides a basis for ROI mask.

**binSize:**

The binning will be done from binSize x binSize pixels to 1 pixel in each panel.

Set binSize to 1 to disable binning.

**tolerance:**

The absolute error bound that will be used during vanilla SZ compression step. So, this is error bound on the binned data.

**NOTE:** Just as a reminder, the absolute error bound is only guaranteed on the binned values. Since binning maps each pixel to a local average, there is no guaranteed error bound on the decompressed data on a pixel-by-pixel basis. This is due to the fact that local averages may be significantly different than the individual pixel values.

**roiM:**

This is the ROI Mask. Used to determine which pixels will be saved lossless.

Consists of nEvents x 32 x 185 x 388 uint8\_t.

roiM is generated by:

Copy calibPanel into each panel in roiM array.

(calibPanel is replicated by nEvents x 32 times)

Use peaks array and peakSize to include peak regions into roiM

**roniM:**

Stands for “Region of Not Interest Mask”. Currently unused.