SignalStorageToolbox

User guide

What is the SignalStorageToolbox ?

Signal storage is a toolbox to transfer the output of simulations into java and matlab. It has been implemented with large digital circuits such as memories in mind. It provides the following features:

- Quickly retrieve signals by name, even in simulations with many signals (millions of signal names)
- Selective loading of signals based on regular expressions
- Java code to load signals : faster than typical matlab implementations for most storage types
  - Fast signal toolbox with many features
- A common client view for a wide range of different simulation output formats. This enables better reuse of postprocessing scripts.
- At this point, the following formats are supported:
  - Spectre PsfAscii (both sweep and DC)
  - Spectre UWI – compressed and non-compressed (transient simulation only)
  - ultrasim UWI – compressed ! (transient simulation only)
    - compressed header allows efficient operation, even with huge netlists
  - hspice & eldo CSDF
  - The effort to implement additional file formats is fairly limited.
- A simple matlab-based Wave Viewer is provided
  - see ./examples/exampleSignalStorageDatabase/signalStorageInfo.m
- UWI stands for Ultrasim Waveform Interface. This interface is implemented in ultrasim and spectre. It allows to provide your own (c/c++) plugin code to write your own fileformat. Two such modules have been implemented:
  - compressed : stores for each signal only the relevant (x,y) values. This allows for very compact data files for simulations of huge, mostly idle circuits such as memories. It allows very fast retrieval of specific signals from disk.
  - noncompressed : stores each signal value at each timepoint. This provides good accuracy, but files are typically larger. Retrieval of specific signals from disk is relatively slow.

Table of Contents

SignalStorageToolbox........1
What is SignalStorageToolbox ?........1
Examples........3
File Formats........4
- PsfAscii (Spectre)........4
- UWICompressed (Spectre and ultrasim)........4
- UWINonCompressed (Spectre)........6

Java functions........8
Matlab functions........9

Examples

Please take a look at the examples in ./examples/ . An example is provided for each of the supported file formats, and some of the noteworthy features are demonstrated in ./examples/features.

For each example,

- go to the corresponding folder.
- Execute ./executeSpectre.sh . (After sourcing the latest version of cadence_mmsim.rc) .
- Take a look at the generated output files.
- Start matlab. If you start matlab in a folder which has a java.opts file, as in the example folders, more memory is made available to java. Matlab's default value is very low.
- Excute makePlot.m (or showFeatures.m). This will give an error message.
- Add the matlab scripts to the path. The following command adds the given folder and its subfolder to the matlab path. You might want to use an absolute path instead for actual use...
    - `addpath(genpath(''../../matlab''))`
- Add the java code to the java path
    - `clear java`
    - `javaaddpath( '../../java/SignalStorage.jar')`
- execute the script makePlot.m (or showFeatures.m) again. This should give some interesting plots.

There is also a limited matlab-based wave browser/plotter. See ./examples/exampleSignalStorageDatabase/signalStorageInfo.m for information.

File Formats

Normally, you would use the following simulator – output format settings:

- ultrasim
  - transient : use UWIultrasim.sh, which generates a compressed output format and a compressed header. Point selection is handled by ultrasim, not by the UWI plugin.
- spectre
  - transient (also MC) : use UWIspectre.sh, which generates a non-compressed output format and a compressed header.
  - transient (also MC) : in case only information about crossings is needed, you can use UWIspectreCompressed.sh, check the settings in the shell script first. You will notice very unpleasant artefacts of point removal, but if you know what you are doing, it might be useful.
  - DC / sweep : use PsfAscii
- hspice
  - transient ASCII

PsfAscii (Spectre)

PsfAscii is the ascii version of the parameter storage format. It is actually not intended to store waves, but it can be used for this anyway. The file format is slightly different between a single DC value on the one hand and the DC sweeps and transient simulations on the other hand, so for now there are two different matlab/java methods to read them.

- Add the following line to the netlist to enable this output format
  - option1 options rawfmt = psfascii

Enabling and configuring UWI output

Use the scripts UWIultrasim.sh and UWIspectre.sh . You will have to update them according to your system configuration and your preferences. The scripts allow to override the default settings by setting environment variables before calling this script. Copy these scripts to your executable $PATH and update them (very well documented).

Feature: with UWI_TO_HARDRIVE=yes, you can easily redirect the output files to the corresponding folder on your local harddrive. Just take a look at the scripts.

UWICompressed (Spectre and ultrasim)

This is a MICAS-made fileformat. It features a separate header file .HDR and a wavefile. Four types of wavefile can be used:

- extension = COA ; ASCII
- extension = COH ; half precision – don't use except if you know what you are doing
- extension = COF ; float precision – good overall choice
- extension = COD ; double precision – if you really don't want to compromise on accuracy

To force spectre to use this outputformat,

- Add the shell scripts in ./scripts to the bash executable path
- use the following kind of command line:
  - UWIspectreCompressed.sh file.scs

To make ultrasim use this format, add these lines to the netlist:

- .usim_opt uwi_lib = ../../libraries/compressed_UWI32.so
- .usim_opt wf_format=COF
- set the environment variable UWI_FORMAT to the same value as wf_format !!!

You can alter some format settings in the shell script:

- Location of the UWI plugin libraries
    - export UWI_BASEPATH=~/projects/spectreBinaryWaves
    - export UWI_LIB_PATH=${UWI_BASEPATH}/libraries
- platform: 32 or 64 bits. If you use the 32 bit spectre version, even on a 64 bit machine, you should also use the 32 bit plugin.
    - export UWI_PLATFORM=32 # 32 or 64
- Header type
    - export UWI_HEADER=H # print header (plot name and wave definitions)
        - a normal, flat header
    - export UWI_HEADER=CH # print compressed header
        - use a hierarchical header. For many circuits, the resulting file will be much smaller than the corresponding flat file, and will hence be much faster to load
    - export UWI_HEADER=NH # do not print header
        - Don't print a header. This can safe some time when doing many identical simulations, as in a sweep or a MC simulation. However, you cannot use this option with a compressed header
- The folder to which the simulation output is written
    - export UWI_OUTPUT_MAP=UWI/
    - export UWI_OUTPUT_MAP=UWI/prefix
    - export UWI_OUTPUT_MAP=~/simulations/UWI/
- wavefile format
    - export UWI_FORMAT=COA #ascii
    - export UWI_FORMAT=COH #half precision
    - export UWI_FORMAT=COF #float (single precision)
    - export UWI_FORMAT=COD #double precision
- compression settings
    - For spectre, use compression settings as desired, e.g.
        - export UWI_ISTEP=0.1e-6
        - export UWI_VSTEP=0.01
        - export UWI_TSTEP=50e-12
            - don't bother about UWI_TSTEP, it doesn't work well
        - With spectre and compression, you will get some strange artifacts in the waveforms. Usable, yet sometimes somewhat confusing.
    - For ultrasim, let ultrasim handle the compression:

        - export UWI_ISTEP=0
        - export UWI_VSTEP=0
        - export UWI_TSTEP=1000e-12

Known limitations:

- All data is kept in memory during simulation. File is only written after simulation completes
    - Potential for out-of-memory
    - No on-the-fly wave updates possible

Strong points:

- Loading a small subset of signals is very fast
- Very small output format
- only option for ultrasim!

UWINonCompressed (Spectre)

This is a MICAS-made fileformat. It features a separate header file .HDR and a wavefile. Four types of wavefile can be used:

- extension = NCA ; ASCII
- extension = NCH ; half precision – don't use except if you know what you are doing
- extension = NCF ; float precision – good overall choice
- extension = NCD ; double precision – if you really don't want to compromise on accuracy

To force spectre to use this outputformat,

- Add the shell scripts in ./scripts to the bash executable path
- use the following kind of command line:
    - UWIspectreNonCompressed.sh file.scs

To make ultrasim use this format, add these lines to the netlist:

- .usim_opt uwi_lib = ../../libraries/noncompressed_UWI32.so
- .usim_opt wf_format=NCF
- set the environment variable UWI_FORMAT to the same value as wf_format !!!

You can alter some format settings in the shell script:

- Location of the UWI plugin libraries
    - export UWI_BASEPATH=~/projects/spectreBinaryWaves
    - export UWI_LIB_PATH=${UWI_BASEPATH}/libraries
- platform: 32 or 64 bits. If you use the 32 bit spectre version, even on a 64 bit machine, you should also use the 32 bit plugin.
    - export UWI_PLATFORM=32 # 32 or 64
- Header type
    - export UWI_HEADER=H # print header (plot name and wave definitions)
        - a normal, flat header

    - export UWI_HEADER=CH # print compressed header
        - use a hierarchical header. For many circuits, the resulting file will be much smaller than the corresponding flat file, and will hence be much faster to load
    - export UWI_HEADER=NH # do not print header
        - Don't print a header. This can safe some time when doing many identical simulations, as in a sweep or a MC simulation. However, you cannot use this option with a compressed header
- The folder to which the simulation output is written
    - export UWI_OUTPUT_MAP=UWI/
    - export UWI_OUTPUT_MAP=UWI/prefix
    - export UWI_OUTPUT_MAP=~/simulations/UWI/
- wavefile format
    - export UWI_FORMAT=NCA #ascii
    - export UWI_FORMAT=NCH #half precision
    - export UWI_FORMAT=NCF #float (single precision)
    - export UWI_FORMAT=NCD #double precision

Strong points:

- No size limitation
- Relatively efficient binary file

Disadvantages:

- Larger filesize then compressed
- slower selective signal loading

Java functions

To be added

Matlab functions

To be added