Sustainable AMS Software Development

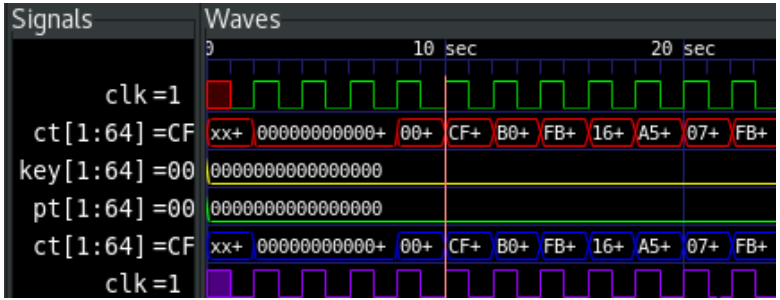
A C++ waveform parser with bindings for several languages

Kleinmeier Benedikt
December 20, 2021

Context

- **Motivation:** Many EDA tools with a plethora of different output formats (FSDB, HDF5, PSF, Scope, ...)

cadence Mentor Graphics SYNOPSYS HDF



- **Goal:** Parser for different waveform files:
  - Supported formats: FSDB, HDF5, PSF and Scope
  - Only support binary data
  - API:
    - getSignalNames()
    - getSignalUnits()
    - getSignals(<optionalSignalList>)
  - Optional name translation to/from SPICE
  - Interfaces to scripting languages by using SWIG

Example usage for C++ (see also live demo)

1

```
auto parser = ScopeParser("test_file.tr.p1");
parser.getSignalNames();
// Returns:
X1/XSUB/DUMMY
...
```

2

```
auto parser = ScopeParser("test_file.tr.p1");
parser.getSignals()
// Returns:
Data in row 0:
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Data in row 1:
  1e-11 9.27613e-10 2.33455e-13 ...
```

Note: getSignals() returns (x,y) or (x,y,z) tuples for each signal!

3

```
auto parser = ScopeParser("test_file.tr.p1");
parser.getSignals({"X1/XSUB/DUMMY"})
// Returns:
Data in row 0:
  0 0
Data in row 1:
  1e-11 9.27613e-10
```

Current status

Finished:

- Parser library is fully implemented in C++ 17
  - with unit tests
  - and CI/CD setup
- Working Perl and Python bindings

Urgent TODOs (see also TODO.md in repo):

- GUI integration
- Performance improvement in PSF code:
  Replace naive “v.push_back(data)” approach by pre-allocating memory, i.e. “v = vector(pre_calculated_size) ... v[i] = data”
- …

Backup slides

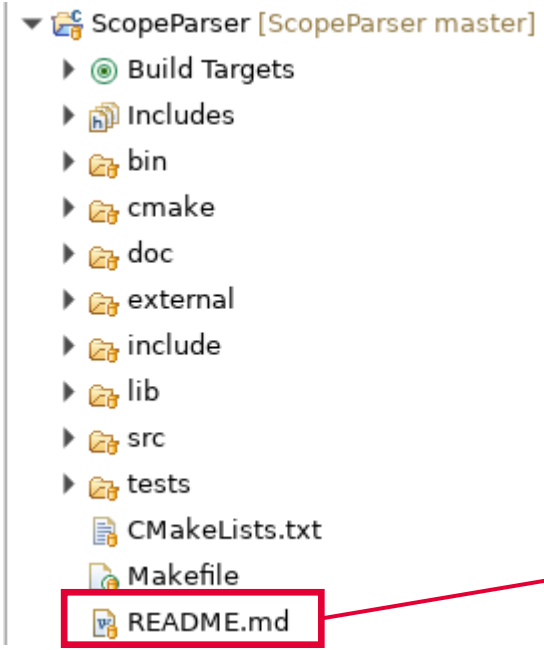
Used tools (as reference for other projects)

- IDE: Eclipse 2021-06
- Build tools:
  - CMake 3.14
  - GNU make
  - GCC 8.2.0
- Unit tests: Catch2 framework plus code coverage with Lcov
- CI/CD
  - Jenkins: https://jenkins.acme.com/job/Waveform%20Parser/
  - SonarQube: https://sonar.acme.com/dashboard?id=CI-WaveformParser





Directory structure



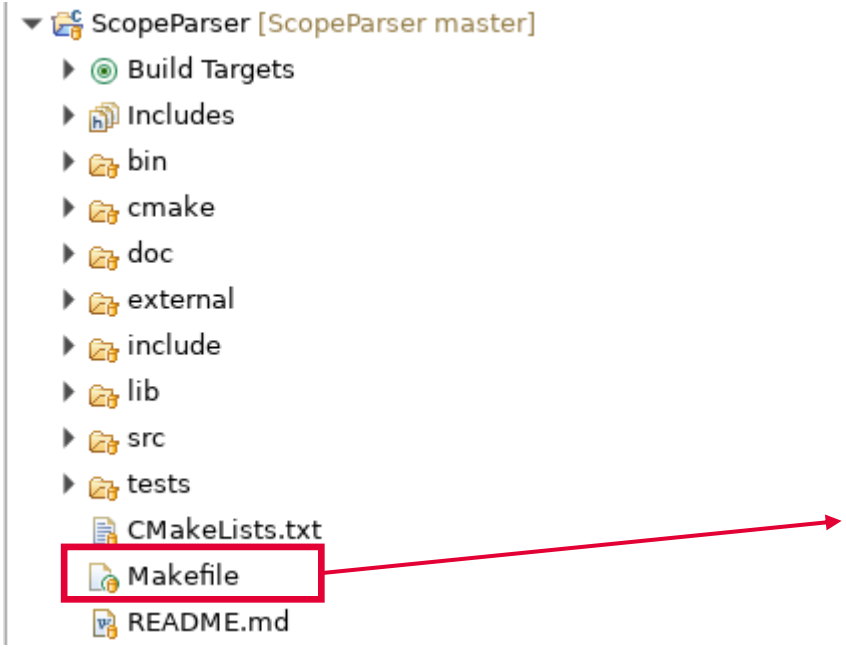

```
# README

This project aims to provide C++ parsers for different EDA simulator formats with a unified interface that provides
several methods like `getSignalNames()`, `getSignalUnits()` and `getSignals()`.

Each signal consists of a (x,y) tuple or, if the signal is complex, a (x,y,z) tuple:

```

Build system: CMake and a Makefile wrapper

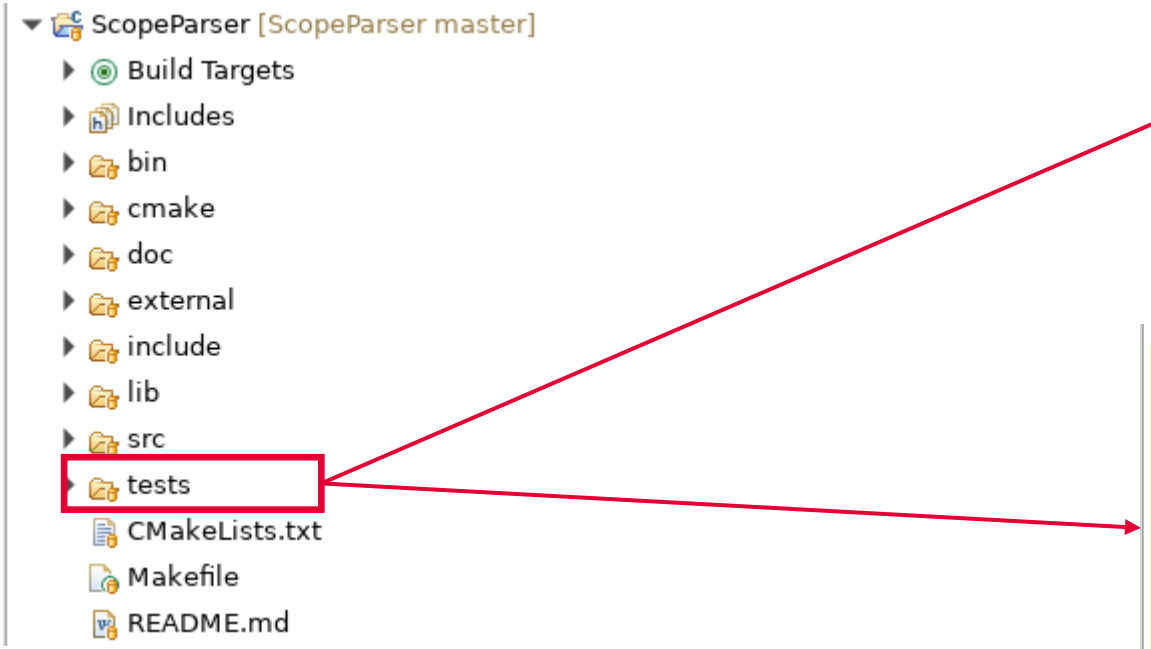



```
# This Makefile configures the build environment by adapting the PATH
# environment variable and delegates the further build to CMake.

################################################################################
# OS Detection Variables
################################################################################
architecture := $(shell uname --machine | cut --delimiter _ --fields 2)
distribution_major_version := $(shell lsb_release --release --short | cut --delimiter . --fields 1)'
os := $(shell uname --kernel-name | tr '[:upper:]' '[:lower:]')
os_description := $(os)$(distribution_major_version)0_$(architecture)

################################################################################
# Build Variables
################################################################################
```

Unit tests and code coverage



LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level				
Test:	coverage.info	**Lines:**	219	257	85.2 %
Date:	2021-07-23 13:01:51	**Functions:**	43	49	87.8 %

Directory	Line Coverage		Functions	
src	83.5 %	177 / 212	88.6 %	31 / 35
src/helper	93.3 %	42 / 45	85.7 %	12 / 14

Generated by: LCOV version 1.14

	Line data	Source code
1		/**
2		* Includes (local to system headers -> alphabetically)
3		**/
4		#include "Parsing.hpp"
5		
6		#include <regex>
7		
8		/**
9		* Constants, Local and Global Variables (if necessary)
10		**/
11		
12		/**
13		* Constructors
14		**/
15		
16		/**
17		* Class Methods
18		**/
19		
20		/**
21		* Non-Class Methods
22		**/
23	256	void toUpperCase(std::string& string) {
24	2144	std::for_each(string.begin(), string.end(), [](char& c){ c = ::toupper(c); });
25	256	}
26		
27	0	void toLowerCase(std::string& string) {
28	0	std::for_each(string.begin(), string.end(), [](char& c){ c = ::tolower(c); });
29	0	}

Uniform file structure

```
#ifndef ${include_guard_symbol}
#define ${include_guard_symbol}

/*******************************************************************************
 * Includes (local to system headers -> alphabetically)
 ******************************************************************************/
${includes}

/*******************************************************************************
 * Aliases, Typedef, Struct and Enum Declarations
 ******************************************************************************/

/*******************************************************************************
 * Class Declarations
 ******************************************************************************/
class ${file_base} {
public:
    // Constants and Variables (static to non-static)
    // Constructors
    // Getter/Setter
    // Operators
    // Methods
private:
    // Constants and Variables (static to non-static)
    // Methods
};

/*******************************************************************************
 * Method Declarations
 ******************************************************************************/
${declarations}

/*******************************************************************************
 * Inline Method Definitions
 ******************************************************************************/

/*******************************************************************************
 * Template Method Definitions
 ******************************************************************************/

 #endif
```

1

2

3

4

5

6

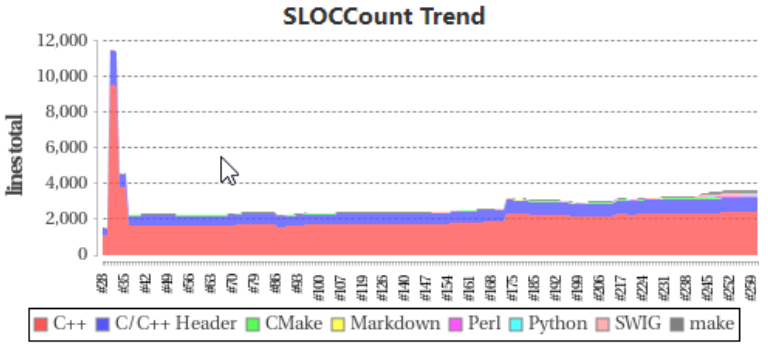
Continuous integration

Stage View

	Declarative: Checkout SCM	Clean	Test: Unit Tests	Coverage: Line and Function	Build: libwaveformparser and C++ example	Analyze: Memory Leaks and Lines of Code	Analyze: SonarQube	Declarative: Post Actions
Average stage times: (Average full run time: ~5min 40s)	1s	1s	1min 26s	2min 20s	53s	27s	27s	802ms
#204 Nov 23 08:29 1 commit	1s	1s	1min 24s	2min 19s	49s	27s	26s	781ms
#203 Nov 22 12:07 1 commit	1s	1s	1min 22s	2min 12s	49s	27s	25s	1s
#202 Nov 22 10:43 1 commit	1s	1s	1min 27s	2min 17s	50s	26s	26s	843ms
#201 Nov 17 14:55 1 commit	993ms	1s	1min 31s	2min 24s	50s	28s	27s	684ms

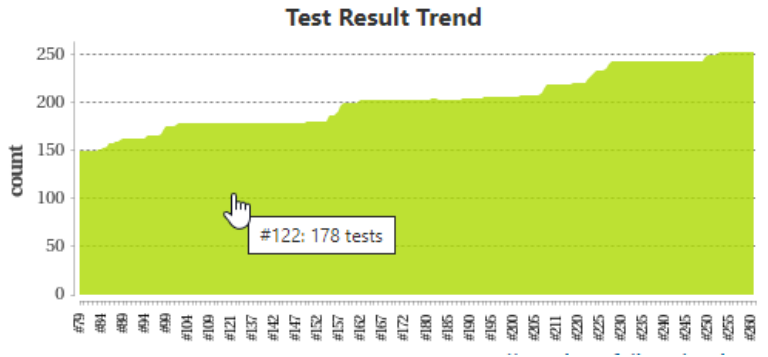
Some metrics

- **Size of libwaveformparser.so:** 3.2 MiB
- **Lines of code:** ~3.5k (new C++) vs. ~11.5k (old C++ code)
- **Code coverage:**
  - Lines: 92.2% (1480 of 1605)
  - Functions: 94.1% (206 of 219)



SLOCCount Results

Language	Lines	Lines Delta	Comments	Comments Delta	Files	Files Delta
C++	2,351		660		24	
C/C++ Header	799		1,675		29	
SWIG	150		51		6	
make	133		14		6	
Perl	90		51	+1	4	
CMake	40		0		9	
Python	8		2		2	
Markdown	6		0		1	
Total	3,577		2,453	+1	81	



Git tags (git tag -n)

Name	Comment
0.1	First working version which can read binary Scope files and which reads all signals in a buffered fashion row-wise
0.2	Replace row-wise and buffered reading by "use seekg()" to extract only certain signals
0.3	Extract signals as (x,y,z) tuple instead of (y,z) tuple, where
0.4	First working version of FSDB parser which relies on Synopsis' low-level API
0.5	First working version of PSF parser which relies on the low-level API under "external/psf"
0.6	First working version of HDF5 parser which relies on the pre-installed HDF5 library
0.7	First working version of fully functional Perl bindings which are generated with SWIG
0.8	First working version of fully functional Python bindings which are generated with SWIG

Resources

- Icons:
  - https://www.gnu.org/graphics/empowered-by-gnu.svg
  - https://commons.wikimedia.org/
- Git repo: https://bitbucket.acme.com/projects/waveformparser/browse
- Jenkins: https://jenkins.acme.com/job/Waveform%20Parser/
- SonarQube: https://sonar.acme.com/dashboard?id=CI-WaveformParser