SMECC – A SME-C compiler using ROSE

Vincent Lanore

December 10, 2012

1 Presentation

SMECC (for SME-C Compiler) is a C99/C++ compiler able to process SMECY pragmas to map function calls to hardware accelerators. SMECC is written using ROSE [**?**, **?**], a tool to write source-to-source translators. First, input code is parsed using ROSE front-end (depends on the input language) into a SageIII AST (ROSE's AST). Then, SMECY pragmas are processed and translated into calls to the SMECY API. Finally, the ROSE backend is called to produce C code with calls to the SMECY API which can be compiled using a regular compiler.

2 Features

SMECC currently supports the following features :

- translation of `#pragma smecy map` directives applied to function calls of the form `function(parameters);`, `varName = function(parameters);` or `type varName = function(parameters);`;
- support from following `arg` clauses : type (in, out...), size and range;
- verification of the contiguity of the vector arguments in memory;
- computing ranges to get the actual dimension of any argument, printing warning when arguments with dimension > 1 are used as vectors;
- automatically finding the size of arrays if not specified in pragma.

3 How to use

Environment Before using the compiler a few environment variable should be set.

- add SMECC directory to the `$PATH` :

```
export PATH=smecc_directory/:$PATH
```

- set `SMECY_LIB` to the directory containing the SMECY library :

```
export SMECY_LIB=smecy_lib_directory/
```

Usage SMECC works mostly like a regular C/C++ compiler. Most C/C++ usual compiler flags will work with a few exceptions and additions (see below). By default, it will *not* compile smecy pragmas (see below).

Specific flags SMECC supports some specific flags. Here are a few examples, for a more complete list type `smecc --help`.

- `-smecy` triggers smecy pragmas translation/compilation; if pragmas contain many expressions SMECC may produce a lot of output: `>\dev\null` is recommended to discard them;
- `-smecy-accel` asks for the generation of the accelerator parts, mainly by outlining the `map`-ped function;
- `--smecy_lib=smecy_lib_directory/` can be used to specify the path to the SMECY library; if specified it will be used instead of the environment variable `SEMCY_LIB`;
- `-std=c99` should be used when compiling C99;
- `-c` will only translate input file instead of compiling it; with input file `fileName.C`, SMECC will generate a `rose_fileName.C` file with calls to SMECY API instead of SMECY pragmas;
- `-fopenmp` triggers OpenMP pragmas compilation using the back-end compiler.

Example To compile a C99 input file with smecy and OpenMP pragmas without useless output type:

```
smecc -std=c99 -fopenmp -smecy input.c
```

4 Known bugs and limitations

Features not yet implemented:

- FORTRAN support;
- only toy implementation of the SMECY API.

AstRewriteMechanism bugs:

- crash if the C++ input file has certain extensions (like ".cpp"), changing the extension to ".C" seems to solve the problem;
- the parser called for the strings is always in C++ mode (not C), commenting out a few lines in a ROSE header prevents front-end errors;
- the parsing of expressions is extremely slow (several seconds to parse ten expressions) and generates 1 file per expression to parse.

Compatibility with ROSE OpenMP lowering

- ROSE OpenMP built-in support conflicts with smecy lowering and requires special handling;
- OpenMP files lowered using XOMP library require special linking, see in `rose_install_dir/src/midend/programTransformation/ompLowering/` for the library files.

Other bugs

- if `-smecy` is not set, multi-line pragmas will lose their \ and fail to compile.

References