tcfggen user manual

Title	tcfggen (Task control flow graph extraction MachSUIF pass)
Author	Nikolaos Kavvadias 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014
Contact	nikos@nkavvadias.com
Website	http://www.nkavvadias.com
Release Date	14 October 2014
Version	1.1.0
Rev. history	
v1.1.0	2014-10-14 Updated header comments in all source files. Added File Listing section in README; added AUTHORS. Updated for Github.
v1.0.0	2014-02-24 Changed documentation format to RestructuredText. Added ChangeLog in separate file.
v0.2.0	2006-06-27 Fixed problem with parsing input arguments. All options: "-lut","-vcg","-fsm","-cac" should work now.
v0.1.0	2006-06-07 Initial release. Code clean-up, added code generation options.

1. Introduction

`tcfggen` is an analysis pass built to be used with the SUIF2/MachSUIF2 compiler infrastructure. `tcfggen` performs (natural) loop analysis in order tomap the control flow of a given optimization unit (i.e. a procedure in the input program) to its task control flow graph (TCFG). It is also used to pass the static information for the loops in the given procedure to the subsequent stage(s) in the form of pseudo-instructions. These pseudo-instructions pass information regarding:

a) the task transitions

b) the points for task entry and task exit

c) the loop parameters (loop bounds and stride) and the basic induction register

d) which instructions should be removed for ZOLC execution

This pass works for the SUIFrm instruction set and has been tested with MachSUIF 2.02.07.15.

2. File listing

The `tcfggen` distribution includes the following files:

/tcfggen	Top-level directory
AUTHORS	List of `tcfggen` authors.
LICENSE	The modified BSD license governs `tcfggen`.
README.html	HTML version of README.
README.pdf	PDF version of README.
README.rst	This file.
VERSION	Current version of the project sources.
lcugen.cpp	Originally part of the loop count unit generator tool (`lcugen`) it used here for making a single pass on the natural loop analysis results in order to assign proper addresses to the data processing tasks (DPTs) of the algorithm under analysis.
lcugen.h	C++ header file containing declarations and prototypes for the above.
rst2docs.sh	Bash script for generating the HTML and PDF versions of the documentation (README).
tcfggen.cpp	Implementation of the `tcfggen` analysis pass.
tcfggen.h	C++ header file containing declarations and prototypes for the above.
suif_main.cpp	Entry point for building the standalone program `do_tcfggen` that implements the pass.
suif_pass.cpp	Define the SUIF pass built as the dynamically loadable library `libtcfggen.so`.
suif_main.h	C++ header file for the above.

3. Technical information

This pass uses the `machine`, `cfg` and `cfa` libraries of MachSUIF. It also depends on the `suifrm` backend. It first generates the natural `loopanalysis` report for the procedure. If formatted for text output, this information would be as follows:

```
Loop info:
  node depth begin end exit
   int: int   Y/N  Y/N  Y/N
   ..................................
```

where:

-node	the number of the corresponding basic block (integer)
-depth	the loop nesting depth (integer)
-begin	a boolean flag to report if a loop begins at the specified node
-end	a boolean flag to report if a loop ends at the specified node
-exit	a boolean flag to report if an exit from the loop is possible from that node.

Based on the loop analysis results, four different types of pseudo-instructions are generated. Their assembly formats are shown below:

```
ldst     <current-taskid>, <next-taskid>, <next-ttsel>, <next-loop_a>
dpti     <id>, <first-bb>, <last-bb>
loop     <loop_a>, <rindex>, <initial>, <step>, <final>
overhead <state>
```

An LDST instruction incorporates all the information needed for creating a task selection LUT entry: for a given task encoding (`current-taskid`), the succeeding task (`next-taskid`), its type (`next-ttsel`) and loop address (`next-loop_a`) are given. These pseudos are attached to the last assembly instruction in the task.

DPTI instructions denote the first (`first-bb`) and last basic block (`last-bb`) of a task (its enumeration given by `id`). This pseudo is attached to the first instruction of this data-processing task.

A LOOP provides a given loop address (`loop_a`), the actual loop index register (`rindex`) and the loop parameters (`initial`, `step`, `final`). This pseudo is attached to the last instruction in the basic block (typically a BLT = *branch if less than*).

An OVERHEAD marks its following non-pseudo instruction whether it must be kept (`state=0` which is the default), replaced by a no-operation (`state=1`), or entirely removed (`state=2`). These pseudos are attached to the specific instructions.

4. Installation

Unpack the `tcfggen` archive wherever you like, e.g. in `$MACHSUIFHOME/cfa/tcfggen`. You don't need to modify anything in the Makefile, if you have a working MachSUIF 2 installation.

The program binary (`do_tcfggen`) will be installed at $NCIHOME/bin and the shared library (`libtcfggen.so`) at `$NCIHOME/solib`, where NCIHOME is the SUIF 2 top-level directory.

5. Usage details

The pass accepts an input file in CFG form to operate. The output file is a SUIF CFG containing the input CFG with the generated pseudo-instructions.

Usage synopsys:

```
$ do_tcfggen [options] input.cfg output.cfg
```

where options can be one (or more) of the following:

-proc <opt-unit> specify the name of the procedure to perform TCFG construction and generation of pseudo-instructions

-lut generate the VHDL source for the task selection LUT

-vcg visualize the TCFG in VCG format

-fsm generate the VHDL source for an FSM implementation of the task selection unit

-cac generate C simulation code for the initialization of the task selection unit.

6. Known limitations

1. MachSUIF (2.02.07.15) only includes natural loop analysis.
2. Currently, there is support for static loops only.
3. An 'optimization unit' can only be a single function or procedure.