ALaRI Tools Howto

Alberto Ferrante

October 25, 2012

This document explains how to run the tools installed at ALaRI.

How to Cross Compile for the PISA Architecture

A PISA cross-compiler is available in the *opt/wattch_alari/bin/* directory. To compile with that cross-compiler:

1. Type

   ```
   . /opt/wattch_alari/start_wattch_gcc
   ```

   (just the 1st time: it is needed to set the environment variables).

2. Type (where the source file is located)

   ```
   /opt/wattch_alari/bin/gcc source_file -o outfile
   ```

How To Run Simplescalar Tools

1. Compile for the PISA architecture (see previous section);
2. type:

   ```
   /opt/simplescalar/pisa/simplesim/<simulator> <compiled_program>
   ```

   where <simulator> can be: sim-fast, sim-cache, sim-profile, and sim-outorder. See simulators help for details on possible parameters.

How To Run Wattch

1. Compile for the PISA architecture (see previous section);
2. type

   ```
   /opt/wattch_alari/bin/wattch <compiled_program>
   ```

   See simulator help for details on possible parameters.

How To Read the Wattch Simulation Results

From the Wattch readme file.

```
SIMULATION AND RESULTS:

Running Wattch should be just like running sim-outorder.  Provide it
with your processor configuration parameters, input binary, and input
data set.  There aren't any additional command line parameters, but
there are some hard-coded parameters that you could change.  For example,
the technology settings (in power.h) and the choice of static vs dynamic
activity factors (power.h).

These are the additional statistics that are generated by Wattch:

1)  Stats that hold the "total power usage" (actually energy) of the
unit over the program execution.  This is basically the summation of
the power usage on each cycle.  We record these stats for each hardware
unit that we model (rename, bpred, etc), each pipestage (fetch, dispatch,
issue), as well as the total power dissipation.  We also record these
stats for each conditional clocking style that we model.  The four
conditional clocking styles currently implemented are:
  a) no conditional clocking -- ie "rename_power"
  b) simple conditional clocking; ie. "rename_power_cc1"
  c) aggressive, ideal (0 power consumed when turned off) conditional
     clocking; ie. "rename_power_cc2"
  d) aggressive, non-ideal (some fraction is still consumed when disabled)
     conditional clocking; ie. "rename_power_cc3"

2)  Stats that hold the average power usage.  This is basically
total_power_usage divided by the number of cycles.

Probably the most interesting clock gating style is cc3.  If you just
want to get one power number and don't care about unit breakdowns look at
"avg_total_power_cycle_cc3".

3)  Unit access statistics are also generated as well as the max number
of of accesses for the different units.  We also track the maximum
processor power dissipation on any given cycle. (max_cycle_power_cc*)
```