Introduction

Heterogeneous Architecture Configurations Generator for Multi2Sim simulator (HeteroArchGen4M2S)

`HeteroArchGen4M2S`: An automatic generator software for configuring and running heterogeneous CPU-GPU architectures on Multi2Sim (M2S) simulator. This tool runs on top of M2S simulator, it allows us to configure the various heterogeneous CPU-GPU architectures (e.g., number of CPU cores, GPU cores, L1, $L2$, memory (size and latency (via `CACTI 6.5`)), network topologies (currently support 2D-Mesh, customized 2D-Mesh, and Torus networks)...). The output files include the results of network throughput and latency, caches/memory access time, and and dynamic power of the cores (can be collected after running `McPAT`).

`HeteroArchGen4M2S` is free software, which is freely to be redistributed and modified it under the terms of the GNU General Public License as published by the Free Software Foundation (For more details `http://www.gnu.org/licenses`).

`HeteroArchGen4M2S` is written to help you configure M2S easily, but non-warranty and non-mechantability.

> A pdf version of this manual is also available in `HeteroArchGen4M2S.pdf`.

Please cite my tool using this bibtex:

```
@article{HeteroArchGen4M2S,
Author = {Tung Thanh Le},
Journal = {https://github.com/ttungl/HeteroArchGen4M2S},
Title = {{HeteroArchGen4M2S: An automatic generator tool for
    configuring and running heterogeneous CPU-GPU architectures}},
Year = {2017}}

Note, paper reference will be updated soon. Enjoy it!
```

Setup Requirements

1. Currently HeteroArchGen4M2S has been tested on 64-bit platforms:
    - Ubuntu 14.04 (final)
2. Required tools to build and run with HeteroArchGen4M2S:
    - Python 2.7
3. Download and install `multi2sim` from:

- `git clone https://github.com/Multi2Sim/multi2sim.git`

4. Download `McPAT` (current version-1.3) from:
   - `https://code.google.com/archive/p/mcpat/`.
   - Unzip it, compile it using `make`, and then go back under `multi2sim` directory, use `mv mcpat/* .` to copy all files from mcpat folder to under multi2sim directory. Now, you are ready to use McPAT, using `./mcpat -h` to check it out.

5. Download and install `CACTI6.5` from:
   - `http://www.hpl.hp.com/research/cacti/cacti65.tgz`.
   - CACTI 6.5 is used to obtain the cache and memory latency.

6. Required benchmarks to run:
   - Download benchmarks. Note that, type the commands under the installed multi2sim directory.

     ```
     git clone https://github.com/Multi2Sim/m2s-bench-splash2.git

     git clone https://github.com/Multi2Sim/m2s-bench-spec2006.git

     git clone https://github.com/Multi2Sim/m2s-bench-parsec-3.0.git

     git clone https://github.com/Multi2Sim/m2s-bench-cudasdk-6.5.git

     git clone https://github.com/Multi2Sim/m2s-bench-heteromark.git
     ```

   - Compile the benchmarks following the README file.

     Note: All the packages are compiled to binary. If you want to modify the benchmarks, you can download the source codes from `https://github.com/Multi2Sim`. In case you want to run CUDA benchmarks, you can download other benchmarks for CPU-GPU systems such as Rodinia, Parboil, etc. Your desktop should have a NVIDIA graphic card (e.g., NVIDIA Quadro 4000), and you need to install the graphic card driver for running the simulation. (When compiling benchmarks, use `-m32` flag after `gcc` to make compatible with `multi2sim` 32-bit (no support 64-bit at this time)).

Download HeteroArchGen4M2S

```
git clone https://github.com/ttungl/HeteroArchGen4M2S.git
```

Build configuration files with HeteroArchGen4M2S

Let's assume you are in the home directory (`$multi2sim/HeteroArchGen4M2S`)

Where are the configuration files?

- Run `/multi2sim/HeteroArchGen4M2S$ python create_sim_configs_files.py`.
- The output files will be saved in the `configs` directory.
- `cd configs` >>> the `configs` folder contains four files, including `memconfig`, `netconfig`, `x86_cpuconfig`, and `si_gpuconfig`.

How to run the simulation?

- Previous steps show how to generate the configuration files. By running `create_sim_configs_files.py`, it also generated a shell script file inside `run_simulation_files` folder. The bash file (shell script) has been `chmod 777` for running.
- Go back under `multi2sim` directory.
- Run `./HeteroArchGen4M2S/run_simulation_files/run-bash-sim.sh`.
- This will create the output files which are the results of the simulation.

Where are the output files after simulation?

- `cd results` >>> Note that, `results` folder contains two files at this point, including `pipeline.out`, `mem.out`.
- With `net_report.out` file, it is generated under the `multi2sim` directory (outside of `HeteroArchGen4M2S` folder), you need to copy this file to `HeteroArchGen4M2S/results`.
- Now, there are three files should be in `results` folder, including `pipeline.out`, `mem.out`, and `net_report.out`(just copied).

Demonstration:

How to run multi2sim with HeteroArchGen4M2S ?

Let's use the `radix` example with 16 cores CPUs (`8` x86 CPUs), 16 cores GPUs (`4` Southern Islands GPUs), 4 Memory Controllers, in a 2D-Mesh for demonstration.

> Important: You need to download `parsec` benchmark from `https://github.com/Multi2Sim/m2s-bench-parsec-3.0`, then unzip it under the `benchmarks` folder in `multi2sim` directory for demonstration.

1. Suppose that you already got the cache and memory latencies for your proposed architecture by running `CACTI6.5`.

2. Suppose that you are under the `multi2sim\HeteroArchGen4M2S$` directory:

   - `sudo vim create_sim_configs_files.py` to configure your architecture. This file includes many parameters that need to be configured.
   - For CPU cores, a set of CPU includes two cores. Each core in the set can have its own L1$ (Data&Instr) or it can share the Instruction-L1$ with the other core in that set, by enabling `L1_Inst_shared` flag in the CPU Memory Parameters settings.
   - For GPU cores, a set of GPU includes four compute units. Each two compute units share with one L1, $eachtwoL1$ shares with one L2$.
   - For benchmarks, you need to modify the name of specific benchmark you want to run, and modify the command line of this benchmark and its path in `create_shell_script` file.
   - For network topologies, HeteroArchGen4M2S currently supports three types of network, including `2D-Mesh`, `customized 2D-Mesh`, and `2D-Torus`. For `customized 2D-Mesh`, you need to specify the paths for local links and hybrid links in your network, as well as their linkwidths.

3. After modifying `create_sim_configs_files.py`:

   - Run `\HeteroArchGen4M2S$ python create_sim_configs_files.py` to generate the configuration files.
   - Checking the configuration files in the `configs` folder.
   - A shell script (`.sh`) has also been generated in the `run_simulation_files` folder. The shell script looks like as below.

     m2s –x86-sim detailed –x86-report
     HeteroArchGen4M2S/results/blackscholes_pipeline.out
     –mem-report HeteroArchGen4M2S/results/blackscholes_mem.out
     –x86-config ./HeteroArchGen4M2S/configs/x86_cpuconfig
     –si-sim detailed –si-config ./HeteroArchGen4M2S/configs/si_gpuconfig
     –mem-config ./HeteroArchGen4M2S/configs/memconfig
     –net-config ./HeteroArchGen4M2S/configs/netconfig
     –x86-max-inst 100000000 –net-report blackscholes_net_report.out
     benchmarks/m2s-bench-parsec-3.0/blackscholes/blackscholes
     16 in_4K.txt prives.txt data-small

   - `cd ..` to `multi2sim` directory.
   - `./HeteroArchGen4M2S/run_simulation_files/run-sim-16-CPU-16-\`
     `SouthernIslands-GPU-benchmark-radix.sh`

4. To check the results, all of them are in the `results` folder, including:
   - `radix_mem.out`
   - `radix_pipeline.out`
   - With `net-l2-mm_radix_net_report.out`, it is saved under the `multi2sim` directory, so just copy into the `results` folder.

5. To read the results:
   - Make sure you are in `HeteroArchGen4M2S` directory.
   - Run `python read_results.py`, the total cycles and network performance results are saved in `results` folder, under the names: `radix_totalCycles.out` and `radix_network_performance.out`.

     For `radix_totalCycles.out`:
   - Cycles: 306794
   - Time (seconds): 3.94

     For `radix_network_performance.out`:
   - Network Throughput (MBps): 32366.64
   - Network Latency (cycles): 23.6969

6. To run the network-only mode, in the file `create_sim_configs_files`, you just need to modify the value of `network_only` to `1`. Thereby, now when running `HeteroArchGen4M2S$ python create_sim_configs_files`, the software will generate the bash file with network-only simulation.

7. To get the dynamic power from McPAT. After running the file `create_sim_configs_files.py`, it also generated a sample xml file in `pipeline_xml_for_mcpat` folder under `HeteroArchGen4M2S` directory. This file is named as following `McPAT_hsa_#cores_benchmark_bmname.xml`, e.g., in this case, its name is `McPAT_hsa_16_benchmark_radix.xml`.
   - Make sure you are in `multi2sim` directory.
   - Run the command lines as follows to update the simulation results from the `radix_pipeline.out` (generated by multi2sim) and then you type the path of `McPAT_hsa_16_benchmark_radix.xml`, and type your output xml file for running McPAT.

   ```
   multi2sim$ ./HeteroArchGen4M2S/pipeline_xml_for_mcpat/pipeline_to_xml_mcpat.sh
   ```

   - Then you will see as below, and should follow the guide:

   Type in the pipeline report from multi2sim:

   ```
   HeteroArchGen4M2S/results/radix_pipeline.out
   ```

Type in any template xml file of mcpat:

`HeteroArchGen4M2S/pipeline_xml_for_mcpat/McPAT_hsa_16_benchmark_radix.xml`

Name a new xml file for input of mcpat:

`HeteroArchGen4M2S/pipeline_xml_for_mcpat/McPAT_hsa_16_radix_result.xml`

- After running the `.sh` file, you will see the file `McPAT_hsa_16_radix_result.xml` in `pipeline_xml_for_mcpat` subdirectory.
- Run McPAT:

'multi2sim$ ./mcpat -infile HeteroArchGen4M2S/pipeline_xml_for_mcpat/
McPAT_hsa_16_radix_result.xml -opt_for_clk 1 -print_level 5 >
HeteroArchGen4M2S/pipeline_xml_for_mcpat/mcpat_hetero_16_radix_output.out

- `mcpat_hetero_16_radix_output.out` contains the dynamic power results of the system as follows.

```
McPAT (version 1.3 of Feb, 2015) is computing the target processor...


McPAT (version 1.3 of Feb, 2015) results  (current print level is 5)
*****************************************************************************
  Technology 45 nm
  Using Long Channel Devices When Appropriate
  Interconnect metal projection= aggressive
  interconnect technology projection
  Core clock Rate(MHz) 3400

*****************************************************************************
Processor:
  Area = 456.669 mm^2
  Peak Power = 455.678 W
  Total Leakage = 84.2285 W
  Peak Dynamic = 371.449 W
  Subthreshold Leakage = 78.0152 W
  Subthreshold Leakage with power gating = 36.2294 W
  Gate Leakage = 6.21323 W
  Runtime Dynamic = 263.559 W

  Total Cores: 16 cores
  Device Type= ITRS high performance device type
    Area = 292.914 mm^2
    Peak Dynamic = 247.926 W
    Subthreshold Leakage = 63.9362 W
```

```
  Subthreshold Leakage with power gating = 29.029 W
  Gate Leakage = 4.98636 W
  Runtime Dynamic = 260.953 W

Total L2s:
Device Type= ITRS high performance device type
  Area = 92.2849 mm^2
  Peak Dynamic = 30.9746 W
  Subthreshold Leakage = 7.38318 W
  Subthreshold Leakage with power gating = 4.20153 W
  Gate Leakage = 0.118424 W
  Runtime Dynamic = 0.00929432 W

Total First Level Directory:
Device Type= ITRS high performance device type
  Area = 14.1044 mm^2
  Peak Dynamic = 38.9904 W
  Subthreshold Leakage = 0.998575 W
  Subthreshold Leakage with power gating = 0.449359 W
  Gate Leakage = 0.177988 W
  Runtime Dynamic = 0.040424 W

Total NoCs (Network/Bus):
Device Type= ITRS high performance device type
  Area = 14.7407 mm^2
  Peak Dynamic = 46.624 W
  Subthreshold Leakage = 4.27901 W
  Subthreshold Leakage with power gating = 1.91074 W
  Gate Leakage = 0.608796 W
  Runtime Dynamic = 0.00902328 W

Total MCs: 4 Memory Controllers
Device Type= ITRS high performance device type
  Area = 22.2144 mm^2
  Peak Dynamic = 3.90557 W
  Subthreshold Leakage = 0.535477 W
  Subthreshold Leakage with power gating = 0.241495 W
  Gate Leakage = 0.0580031 W
  Runtime Dynamic = 0.427028 W

Total NIUs: 4 Network Interface Units
Device Type= ITRS high performance device type
  Area = 16.6974 mm^2
  Peak Dynamic = 2.16692 W
  Subthreshold Leakage = 0.679449 W
  Subthreshold Leakage with power gating = 0.305752 W
```

```
    Gate Leakage = 0.202921 W
    Runtime Dynamic = 1.51684 W

  Total PCIes: 1 PCIe Controllers
  Device Type= ITRS high performance device type
    Area = 3.7129 mm^2
    Peak Dynamic = 0.862065 W
    Subthreshold Leakage = 0.203374 W
    Subthreshold Leakage with power gating = 0.0915182 W
    Gate Leakage = 0.0607388 W
    Runtime Dynamic = 0.603445 W

*****************************************************************************************
Core:
      Area = 18.3071 mm^2
      Peak Dynamic = 15.4953 W
      Subthreshold Leakage = 3.99601 W
      Subthreshold Leakage with power gating = 1.81431 W
      Gate Leakage = 0.311648 W
      Runtime Dynamic = 16.3096 W
...
...
...
```

- From this file, you can be able to collect all the dynamic power information you need for evaluations.

Now you are ready to go. Happy hacking the code!

Claims:

I would like to thank the open source multi2sim community. This work is also inspired by M2StoMcPAT in Matlab, but I implemented completely in Python, with heterogeneous CPU-GPU architectures.

```
Tung Thanh Le
ttungl at gmail dot com
Release: Version 1.0 (02/18/17)
```