Abstract:

In data IO intensive workloads such as image processing and CNN, it is being increasingly pivotal to increase data throughput and enhance IO delay by increasing the cache locality. However, physical cache's performance is limited by its real-life physical constraints, and researchers have been exploring non-conventional methods to increasing cache locality. One of the methods to achieve better special locality is by Approximate Cache Access methods, exemplified by the "Bunker Cache" which cache controller can exploit the redundancy nature of the image data and map the vertical 'lines' of memory into the same cache address. In that case, whenever the processor wants to access the data 'line' of a cached data adjacent in 2D space, the cache access would become a hit and returns the same data that have just accessed before. One of the drawbacks of the bunker cache was that some extend of ISA level support was required. One of the parameters in the bunker cache controller was the "step size" and proof-of-concept experiments have shown that performing a FFT calculation on the IO data-stream would extract such parameter rather effortless. So, I propose to implement a DSP based co-processor that work together with the Bunker Cache controller to automatically calculate the parameters with each IO on image data from disk to memory, at the same time seamlessly and synchronously computes FFT on the data stream and updates the controller parameter registers to hide the implementation details of the Bunker Cache from the programmer. Currently, this project is still under development and I plan to construct a simplified bunker cache and the co-processor on a home-brew Verilog testbench environment.

One of the original objectives of the project was that I would simulate all behavior of the bunker cache on a simulation platform, and I choose Gem5 as the simulator. However, after playing with Gem5 for a long while I found Gem5's cache simulation system was extremely hard to work with and found no good way to change the cache simulator object's behavior without influencing the functionality of other parts of the code within the testbench program. So, I'm planning to discard the simulator approach and construct everything in Verilog testbench simulation environment, as I have a much clearer and intuitive understanding on how each part of my project interact with each-other.

Till now I have only finished the basic FFT co-processor core with appropriate interface with outside.

Plan for next: the next step is to write a simple bunker cache controller in Verilog with appropriate communication interface with the FFT core. Due to the intrinsic difficulty of running high-level testbench on Verilog; The plan now is to use a high-level language (such as python) to simulate the behavior of the simple bunker cache / co-processor and run comparison on the effectiveness of bunker cache with pre-defined parameter and generated parameters.