GreenLight Technical Report

2009

FPGA Acceleration

Rajesh Gupta: Specialized Co-Processing in a Generalized Execution Environment

Research Activities

We have been exploring architectural alternatives for coprocessing that enable use of the coprocessors in a general purpose programming environment. In particular, we have explored machine organizations that place functional accelerators at various distances from the CPU: ranging from memory-mapped coprocessors to tightly integrated on-chip functional units. A specific organization that has shown promising early results places coprocessors on a CPU-memory bus such that the co-processor operates in a memory-coherent manner with the main processing. We call this approach "coherent coprocessing" (CCP). A coherent coprocessor operates in parallel and in the shared virtual address space as the main processor (or multiple processing elements in case of a multi-core system). This allows the coprocessing units to avoid unnecessary memory-memory copy operations that are often deployed in commonly used asynchronous coprocessors. Also, due to shorter latencies to coprocessing units, it is possible to use acceleration opportunities at finer grain than possible with typical custom computing machines.

To explore design alternatives and quantify gains in performance and power effectiveness, we have classified potential applications into the type of algorithmic core that dominates most of the execution time. This classification is based on the "recognition-mining-synthesis" framework first proposed by Intel [Chen, *et. al.* IEEE Proceedings, 96(5), 2008]. Our current focus is on dynamic programming algorithms that dominate a number of compute intensive applications in the bioinformatics area. In particular, we have focused on MS-Alignment as a part of the Inspect code for computational mass spectrometry. Inspect is a general purpose database search algorithm which efficiently and confidently identifies modified peptides in mass-spectrometry data. Unfortunately, proteins undergo significant post-translational modification (PTM) in order to modulate their structure, regulate their function, and as part of signaling networks. The functions of many post-translational modifications are well characterized. Phosphorylation is used to signal or activate proteins. Methylation and acetylation are used to modify the state of chromatin and influence gene regulation. These are fairly common modifications that can be found in any proteomics sample. However, there are also several well-characterized rare modifications, e.g., actin arginylation and dipthamide [Karakozova 2006, Van Ness 1980]. Hundreds of other modifications are known; recent research points to newly discovered in vivo modifications, as well as many unknown chemical adducts [Tanner 2008, Chen 2007]. In light of these and future discoveries, the ability to identify all modifications in a sample is particularly attractive. An unrestrictive search can confidently identify rare and uncharacterized modifications.

The central challenge of unrestrictive search is speed. In the unmodified search, database search programs filter the database to only consider a subset of the potential peptides as candidates for scoring. Inspect uses sequence tags to filter the database; other programs use a parent mass filter. Regardless, each of these relies on a limited alphabet composing the peptide sequences. There are 20 amino acids used in protein sequences. When users enumerate a modification (e.g., oxidized methionine), a new mass can be added to the 20 standard amino acids. This slightly expanded alphabet can be easily incorporated into filtering techniques. Unrestricted search, however, have a massively expanded alphabet, and cannot readily adopt standard filtering algorithms. Instead of a 20 character alphabet, the unrestricted search considers 20 amino acids, and up to 500 modified versions of each (up to 250 dalton addition or subtraction to the mass of the amino acid). By considering all possible modifications, and without any filtering MS-Alignment is at least two orders of magnitude slower than Inspect.

A slower run time for MS-Alignment is at odds with the trend for increasingly large data sets in proteomics. Large data sets, containing millions or tens of millions of spectra, are now routine. Initial profiling of the code revealed that 99.4% of run time was spent inside a single function, making MS-Alignment an ideal candidate for hardware based acceleration. The function, FreeMod.c::SeekMatch1PTM, utilizes a dynamic programming table to place post-translational modifications within a candidate peptide. This is essentially the search function for MS-Alignment, although it is a complete search of the database, unaided by filtering. We targeted CCP acceleration of the search kernel. Each candidate amino acid string and each possible modification within a substring of that candidate is given a fast score to quantify and limit the candidate matches. Upon completion, the top 100 candidates are passed to the next stage for scoring based on multiple features combined by a support vector machine. The search kernel can be partitioned across input spectra, across proteins in the database, across candidate AA strings, and across possible sub-strings of each candidate AA string. Our CCP architecture uses independent processing elements, each interfaced to coherent memory. By independently streaming data, we avoid large centralized buffers in memory. Thus the search kernel is a tiled array of search modules, each operating on a unique spectra sample. Each search module has a dedicated load/store unit that access to the coprocessor memory. The relatively small input spectra reside in FPGA block RAMs during search. We buffer potential matches on chip in order to quickly profile their scores to establish a threshold and commit only the most relevant matches to memory. The final stage of the search kernel sorts the candidates and returns the best hundred matches using a tree-like reduction in hardware. With eight search modules in each Virtex-5 FPGA for a total of 32 modules running in parallel, our results show a speedup of over 300X for the kernel execution and at least 150X for the entire application when compared to execution on a cluster of 128-node Xeon machines.

Our immediate ongoing work includes parallelization of the search kernel code for execution using GPGPU via CUDA. We expect to complete this implementation by Spring 2009, thus providing us with baseline comparisons of computation efficiencies across three computing fabrics. Our next step would then include organizations where the coherent coprocessing is integrated with the integer pipelines of the main processing units and accessed using extension of instruction set architecture of the main processing units.

For more information please contact Rajesh Gupta, rgupta@ucsd.edu