Summary

In the initial decades of the 21st century, we are approaching the ExaScale era in which computing systems will be able to perform up to $2^{60}(\approx 10^{18})$ FLOPS (i.e. floating point operations per second). It is impossible to reach this impressive performance with homogeneous architectures, since high-performance general purpose processors consume too much energy per computation, no matter whether high computational capabilities are required in data centers or in edge applications. Due to the likely end of Moore's law even for scaling, higher and higher demands for low power and high performance are now satisfied only by heterogeneous computing systems, which are composed by host devices and a set of co-processors with various of capabilities to handle particular tasks.

This thesis focus on the design of FPGA-based accelerators, which exploit a reconfigurable spatial computing architecture to achieve massive parallelism at a very low power consumption. The predominant design methodologies for an FPGA is still based on Register Transfer Level (RTL) models, which are entirely diverse from software models and thus prevent the software and hardware co-design and the principle "Write once, run anywhere''. In this thesis we propose several system level design methodologies for FPGAs via high-level synthesis, which enables the portability of software originally written in C, C++ or OpenCL for CPUs or GPUs to hardware platform.

The performance of an application on a piece of hardware is bounded by two factors: the peak performance of the processor/accelerator and the bandwidth of the memory. The roofline model is used to estimate the peak performance of an algorithm according to its computation-to-communication ratio, to classify the algorithm into computation-bounded, memory-bounded or somewhere in the middle and to analyze the bottlenecks of the algorithm for further optimization. Note that the memory bandwidth of FPGAs has been historically lower than that of GPUs, and FPGAs are typically less efficient than GPUs at double-precision floating point, while FPGAs shine when it comes to on-chip memory bandwidth and fine-grained low precision computations.Thus the classification depends on both the algorithms and the devices.

This thesis covers both computation-dominated and memory-dominated designs.

First we consider financial models as examples of computation-intensive algorithms, namely the Black Scholes model and the Heston model of the prices of one vanilla option (i.e. European vanilla option) and two exotic options (i.e. Asian option and European barrier option) respectively. We optimized and implemented these algorithms on three platform types (i.e. CPUs, GPUs and FPGAs) and a total of five devices. Obviously both FPGAs and GPUs outperformed CPUs significantly. Even an embedded FPGA achieved about 15x better performance than a data center class CPU. The FPGAs achieved 4x to 5x operations per Watt than a GPU fabricated in the same process (e.g. 16nm and 28nm).

Secondly, we proposed to optimize the memory access bandwidth of memory-bounded algorithms, by using pre-designed C++-based inline caches rather than the tedious on-chip local memory design approach. The latter requires designers to manually exploit the memory access patterns and manage the data movements and synchronizations, the on-chip RAM architecture, the kernel interfaces to access external memory, and functional verification. We applied inline caches with different types and configurations to three algorithms from very different application areas such as machine learning, databases, and computer vision respectively. In summary, our cache implementations improved performance by up to 8x and energy by about 2x with respect to the out-of-the-box unoptimized code, achieving comparable results to the best available manual optimizations of the on-chip memory architecture, while requiring a much shorter design time.

Finally, we explored the design space of several types of machine learning algorithms including convolution neural networks and recurrent neural networks. They are both memory and compute-bounded. These models were first designed and trained in a framework such as Tensorflow and then our automation tool generated self-contained C++ projects that are supported by high level synthesis tools. We proposed a dataflow-based acceleration methodology by which we could migrate our designs on various target FPGAs and achieved excellent performance due to the dramatically reduced number of the external memory accesses. We applied this methodology to two neural networks that we designed targeting an embedded FPGA (for edge computations). The first one is a CNN variant named ShiftShuffleNet. The top-1 accuracy is up to 68.5% and top-5 accuracy is up to 88.2% on ImageNet, despite a very heavy quantization with 4-bit activations and 1-bit weights. We designed a configurable ShiftShuffle block where the dataflow paths are configured by the host processors via a set of micro-instructions to implement the full net on a resource-limited embedded FPGA. We not only achieved a competitive accuracy, but also improved the actual inference speed in terms of frames per second (about 96.5 fps). The second is a recurrent neural network based on an LSTM cell. This network was trained by another project in our lab, and had a very good accuracy compared to other feed-forward neural networks. On the embedded FPGA, it achieved a comparable performance and about 2x operations per Watt as a datacenter-class Xeon CPU.

All these experiments prove that high level synthesis is not only mature for both research and commercial uses, but also excellent in terms of the achieved performance and power with respect to CPUs and GPUs. Hence FPGAs can readily be exploited in heterogeneous computing systems for low power and high performance demands in both data centers and edge applications.