Accelerating the CKY Parsing Using FPGAs

Yasuaki Ito (110015)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 14, 2003

Keywords: CKY Parsing, FPGAs, Reconfigurable architectures, Reconfigurable computing.

The main contribution of this paper is to present an FPGA-based implementation of an instance-specific hardware which accelerates the CKY (Cocke-Kasami-Younger) parsing for context-free grammars.

An FPGA (Field Programmable Gate Array) is a programmable VLSI in which a hardware designed by users can be embedded instantly. The user's hardware logic design can be embedded into the FPGAs using the design tools supplied by the FPGA vendor. Our goal is to use the FPGAs to accelerate the CKY parsing. In particular, the challenge is to develop FPGA-based solutions which are faster and more efficient than traditional software approaches.

Parsing find its application in various fields, such as pattern recognition, programming languages, natural language processing, etc. A number of applications could profit if efficient ways to accelerate parsing was available. Real time speech-recognition is an example of such applications.

Our approach to accelerate computations using FPGAs is inspired by the notion of *partial computation*. Let $f(y, x)$ be a function to be evaluated in order to solve a given problem. Note that such a function might be repeatedly evaluated only for a fixed y. When this is the case, the computation of $f(y, x)$ can be simplified by evaluating an instance-specific function f_y such that $f_y(x) = f(y, x)$. Our novel idea is to build a hardware that is optimized to compute $f_y(x)$ for a fixed y and various x. More specifically, our goal is to present an FPGA-based instance-specific solution for problems that involves a function evaluation for $f(y, x)$ satisfying the following properties:

1. The value of a fixed instance y depends on the instance of the problem, and
2. The value of $f(y, x)$ is repeatedly evaluated for various x to solve the problem.

The main contribution of this paper is to present an instance-specific hardware which accelerates the parsing for context-free grammars using the FPGA-based approach described above. Let $f(G, x)$ be a function such that G is a context-free grammar, x is a string, and $f(G, x)$ returns a Boolean value such that $f(G, x)$ returns TRUE iff G derives x. For the purpose of instance-specific solution for parsing context-free languages, we

present a hardware generator that produces a Verilog HDL source that performs the CKY parsing for any given context-free grammar G. The key ingredient of the produced design is a hardware component to compute a binary operator $\otimes_G$ such that $2^N \times 2^N \rightarrow 2^N$, where N is the set of non-terminal symbols in G. More specifically, let U and V be a set of non-terminals in G that derive strings α and β, respectively. The operator $U \otimes_G V$ returns the set of non-terminals that derive $\alpha\beta$ (i.e. the concatenation of α and β). Let n denote the length of the input string that is $n = |x|$. The CKY parsing algorithm repeats the evaluation of $\otimes_G$ for $O(n^3)$ times.

The generated Verilog HDL source is compiled using the Xilinx design tool, and the object file obtained is downloaded into the Xilinx Virtex-II series FPGAs. The programmed FPGA compute $f_G(x)$, i.e. determines if G derives x for a given string x. In our hardware CKY parsing system, given strings $x_1, x_2, x_3, \ldots$ by the host PC, the FPGA computes and returns $f_G(x_1), f_G(x_2), f_G(x_3), \ldots$ to the host.

From the theoretical point of view, our instance-specific solution is much faster than the software solutions. To evaluate the performance of our CKY parsing system, we compared it with traditional software. Traditional software approach evaluates $\otimes_G$ by checking all p production rules in $O(p)$ time. The CKY parsing using the algorithm runs in $O(n^3 p)$ time. On the other hand, our instance-specific solution evaluates $\otimes_G$ in $O(\log b)$ time and the CKY parsing using this approach runs in $O(n^3 \log b)$ time, where b represents the number of non-terminal symbols. Since $b \leq p$ always hold, our solution is faster than these software approaches from the theoretical point of view.

We have evaluated the performance of our instance-specific solution using the Virtex-II series FPGA. In order to evaluate the performance of our instance-specific solution, we also have implemented some software solutions and measured the performance using IBM PC-compatible(Xeon processor). The results show that our instance-specific hardware attains up to 3,000 speed-up factor over the software solutions.