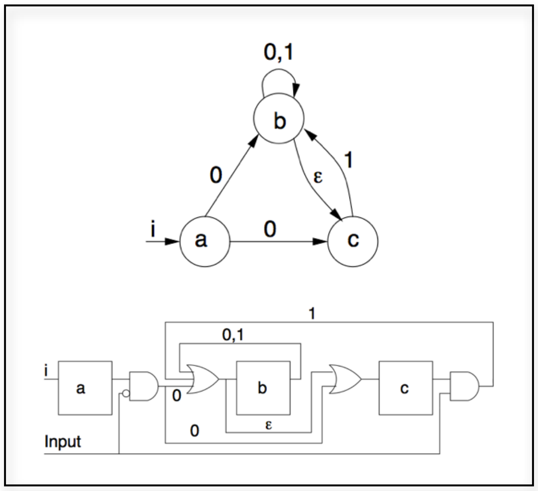
Fast Regular Expression Matching Using FPGAs

Reetinder Sidhu, Viktor K. Prasanna

Year of publication: *2001*
Area: *Applications*



Creating a matcher for a specific regular expression is typically a two-step process. First an NFA is derived from the grammar representing the regular expression. Second, the NFA is converted into a corresponding DFA. The DFA is then typically implemented in software and used to identify strings in an I/O stream.

In this paper, the goal of the authors was to accelerate regular-expression matching by implementing regular-expression matchers in reconfigurable hardware. Their approach was quite novel; they completely skipped the DFA conversion step and directly implemented the NFA in hardware. This reduced the amount of time it took to implement the matcher in hardware. It also avoided a step that, in the worst case, may require exponential time to complete. Their NFA-based matchers require O(n) time and O(n^2) space where n is the length of the regular expression. The NFA-based hardware matcher processes one character per clock cycle.

This is an excellent example of an application paper that demonstrates how reconfiguration not only enables an application, but also provides a new way to look at a problem. Because they are typically developed in the context of general-purpose processors, NFAs are usually just a step on the path toward implementing a DFA-based matcher. However, as demonstrated by this paper and others that followed it, NFAs can be an effective way to implement regular-expression matchers on a reconfigurable fabric. The authors showed that NFA netlists can be directly implemented on FPGAs as one-hot state machines, allowing the state machine to directly represent the set of states in which the NFA might currently be, and they provided a simple algorithm that converts an NFA graph into a netlist that can be placed and routed by vendor place and route tools.

Though the NFA-based compilation process has fewer steps than a DFA-based process, it is still hampered by the need to place and route the NFA-based netlists, a process that can take minutes or longer. As such, place and route time will typically represent the lion's share of total processing time required to convert a regular expression to a hardware matcher. The author's solution to this problem was to introduce a theoretical architecture (SRGA) that made constructive placement simple, but there remains no practical solution to this problem. Still, the results from this paper had immediate practical implications for applications that can cope with place and route overhead and helped to enable the work reported in another paper included in this volume: "Assisting Network Intrusion Detection with Reconfigurable Hardware."

Brad Hutchings

DOI: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1420919