Gene Matching Using JBits[⋆]

Steven A. Guccione[⋆⋆] and Eric Keller

Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 (USA)
{Steven.Guccione ,Eric.Keller}@xilinx.com

Abstract. As the emerging field of bioinformatics continues to expand, the ability to rapidly search large databases of genetic information is becoming increasingly important. Databases containing billions of data elements are routinely compared and searched for matching and near-matching patterns. In this paper we explore the use of run-time reconfiguration using field programmable gate arrays (FPGAs) to provide a compact, high-performance matching solution to accelerate the searching of these genetic databases. This implementation provides approximately an order of magnitude increase in performance while reducing hardware complexity by as much as three orders of magnitude when compared to existing commercial systems.

1 Introduction

One of the fundamental operations in computing is string matching. Here, two linear arrays of characters are compared to determine their similarity. This operation can be found across a wide range of algorithms and applications. One area where string matching has recently received a renewed interest is in the area of bioinformatics, in particular in the area of searching genetic databases.

With the initiation of the Human Genome Project [2] in the early 1990s, the amount of data to be searched, as well as the number of searches being performed on this data has continued to increase. Because of the size and ongoing growth of this problem, specialized systems have been commercially introduced to search these databases of genetic information.

In this paper we present a system used to implement one of the most popular genetic search algorithms, the Smith-Watermann algorithm, using run-time reconfiguration. This approach provides not only smaller, faster circuits, but also reduces the input-output requirements of the system while simplifying hardware / software interfaces.

2 The Smith-Watermann Algorithm

While many applications performing string matching look for an exact match to the searched data, many other applications are interested in finding approximate matches. This requires a somewhat more complex algorithm than the search for exact matches.

[⋆] This work was supported in part by DARPA in the Adaptive Computing Systems (ACS) program under contract DABT63-99-3-0004.

[⋆⋆] Steven Guccione is now at QuickSilver Technology, Inc. steve.guccione@qstech.com

One area where inexact string matching has become important is in the searching of genetic databases. The optimal algorithm for inexact search in the field of bioinformatics is typically known as *Smith-Watermann*[8] and uses a dynamic programming technique. The algorithm compares two strings S and T by performing a pairwise comparison of each element in the two strings, then computing a score to determine the similarity of the two strings. Figure 1 gives a two-dimensional representation of the algorithm. The two strings S and T are compared and intermediate values a, b and c are used to produce the intermediate result, d. This calculation is repeated once for each pairwise element comparison.

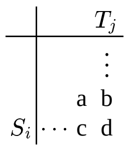

Fig. 1. Pairwise comparisons in the Smith-Watermann matching algorithm.

The matching algorithm itself is given in Equation 1. If the elements being compared are the same, the value a is used to calculate the result value d. If the elements in the two strings are not the same, then the value of a plus some *substitution* penalty is used. The result value d is determined by taking the minimum of this value, the value of b plus some *insertion* penalty and the value of c plus some *deletion* penalty.

$$d = min \begin{cases} \begin{cases} a & \text{if } S_i = T_j \\ a + sub & \text{if } S_i \neq T_j \end{cases} \\ b + ins \\ c + del \end{cases} \tag{1}$$

In the case where string S is of length m and string T is of length n, the algorithm begins by comparing S_0 and T_0 and proceeds onward until a final value of d is calculated at the comparison of S_m and T_n. This value is the *edit distance* between the strings.

3 The JBits™ Implementations

Rather than using the standard VHDL design flow to implement the Smith-Watermann algorithm, the Xilinx® *JBits* toolkit was used [5]. *JBits* was particularly useful in the implementation of this algorithm because there were several opportunities to take advantage of run-time circuit customization.

There are four different opportunities for run-time circuit customization. Three of these are the folding of the constants for the insertion, deletion and substitution penalties into the LUTs. Rather than explicitly feeding a constant into an adder circuit, the constant can be embedded in the circuit, resulting in (in effect) a customized constant adder circuit. Note that these constants can be set at run time and may be parameters to the circuit.

The fourth run-time optimization is the folding of the match elements into the circuit. In genomic databases, a four character alphabet is used to represent the four bases in the DNA molecule. These characters are typically denoted *A* for adenine, *T* for thymine, *G* for guanine and *C* for cytosine. In this circuit, each character can be encoded with two bits. The circuit used to match S_i and T_j does not require that both strings be stored as data elements. In this implementation, the S string is folded into the circuit as a run-time customization. Note that the string values are not fixed constants and will vary from one run to another. This means that the entire string S is used as a run-time parameter to produce the customized circuit.

This design uses a feature of the algorithm first noted by Lipton and Lopresti [7]. For the commonly used constants, 1 for insert/delete and 2 for substitution, b and c can only differ from a by +1 or -1, and d can only differ from a by either 0 or 2. Because of this, modulo 4 encoding can be used, thus requiring only 2 bits to represent each value. The final output edit distance is calculated by using an up-down counter at the end of the systolic array. The up-down counter is initialized to the match string length which makes zero the minimum value for a perfect match.

Further optimizations were performed to more efficiently map the design to the Virtex™ architecture. These optimizations make use of the Virtex carry-chain, which reduced the delay of the circuit since general routing was not needed internal to the processing element. The optimization is evident in Equation 2 which is equivalent to Equation 1 . The equation is basically a wide or gate which is efficiently implementable with the Virtex carry-chain. Another optimization evident from the transformed equation is the fact that d is equal to a or $a + 2$. Because of this, the least significant bits of a and d are always equal. Therefore, only 1 bit is needed to represent d.

$$d = \begin{cases} a & \text{if } b \text{ or } c \text{ equals } a - 1 \text{ or } S_i = T_i \\ a + 2 & \text{if } b \text{ and } c \text{ equal } a + 1 \text{ and } S_i \neq T_i \end{cases} \tag{2}$$

4 Other Current Implementations

As the computing demands of bioinformatics has continued to increase, commercially available solutions to the problem of searching genetic databases have become available. Today, the three major systems used commercially all take different approaches. It should also be noted that these systems all support a variety of matching algorithms in addition to Smith-Watermann. Table 1 gives a comparison of the various technologies currently available to perform Smith-Watermann matching. For a historical comparison, the *Splash 2* work of Hoang has also been included.

5 Conclusions

A gene matching system using run-time reconfiguration and operating on a single FPGA device has been presented. This system is able to perform Smith-Watermann

Table 1. This displays both performance and hardware size for various implementations.

	Processors per Device	Devices	Updates per sec
Celera (Alpha cluster)[1]	1	800	250B
Paracel (ASIC)[3]	192	144	276B
TimeLogic (FPGA)[4]	6	160	50B
Splash 2 (XC4010)[6]	14	272	43B
JBits (XCV1000-6)	4,000	1	757B
JBits (XC2V6000-5)	11,000	1	3,225B

matching at a rate of over three billion matches per second. This compares favorably to the currently available systems used commercially in this field. In the area of performance, the run-time reconfiguration approach provides an order of magnitude increase over both custom ASIC and multiprocessor systems, while reducing the hardware complexity by two to three orders of magnitude.

Such results would often indicate that some system parameter, usually flexibility, has been lost. This, however, is not necessarily true. It is possible to use similar techniques to implement other matching algorithms other than Smith-Watermann using run-time reconfiguration. Interestingly, there may be little or no advantage to implementing sub-optimal matching algorithms using this approach. Because this implementation appears to be limited more by data input / output than by processing power, implementing a faster algorithm may not provide substantial increases in performance. This would make the sub-optimal algorithms much less desirable.

As the field of bioinformatics continues to grow, and various fields from drug design to law enforcement come to rely on this technology, it is expected that interest in high performance matching systems will also grow. Reconfigurable logic and run-time reconfiguration promise to permit faster, less expensive systems to meet these needs.

References

1. Celera Genomics, Inc. World Wide Web site http://www.celera.com/, 2002.
2. Human Genome Project Information. World Wide Web http://www.ornl.gov/hgmis/, 2002.
3. Paracel, Inc. World Wide Web site http://www.paracel.com/, 2002.
4. TimeLogic, Inc. World Wide Web site http://www.timelogic.com/, 2002.
5. Steven A. Guccione, Delon Levi, and Prasanna Sundararajan. JBits: A java-based interface for reconfigurable computing. In *Proc. 2nd MAPLD*, 1999.
6. Dzung T. Hoang. Searching genetic databases on splash 2. In *IEEE Workshop on FPGAs for Custom Computing Machines*, pages 185–191, April 1993.
7. Richard Lipton and Daniel Lopresti. A systolic array for rapid string comparison. In *Chapel Hill Conference on Very Large Scale Integration*, pages 363–376, 1985.
8. T. F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.