FPGA Acceleration of Bioinformatics on the XD1: A Case Study

James Maltby, *Cray Inc*, **Steve Margerm** *and* **Jeffrey Chow**,*Cray Canada*

ABSTRACT: *Bioinformatics algorithms are particularly well-suited for FPGA acceleration because of their natural parallelism and use of subword data types. The Smith-Waterman algorithm is widely used in Bioinformatics and several FPGA-based accelerators have been marketed. This paper will describe our experiences porting the Smith-Waterman algorithm to the CrayXD1Application Accelerator Coprocessor.*

KEYWORDS: XD1, FPGA, Bioinformatics, Smith-Waterman

1. Introduction

The use of FPGAs for the acceleration of scientific applications, also known as Reconfigurable Computing, is an exciting field of current research. However, the characteristics of the application must be well matched to the characteristics of FPGAs and the acceleration platform, in this case the Cray XD1. In this paper we will examine a typical bioinformatics algorithm, Smith-Waterman, and describe how we accelerated it on the Cray XD1. We will examine what issues need to be considered to achieve maximum performance in this system. We will also describe our logical implementation on the FPGA and how it communicates with the host processor.

Because FPGA programs (or "cores") are currently more time-consuming to generate than applications coded in higher-level languages, it is best to select an application that has one or two computation-intensive routines that use the majority of the time in the algorithm. FPGAs are also more well suited to integer, character and subword operations than floating-point operations, because of the large size of floating-point units relative to the current generation of FPGAs. Finally, it is important that the algorithm display a high degree of inherent parallelism to take advantage of the parallelism possible in an FPGA core.

Many bioinformatics algorithms display the characteristics mentioned above. For this reason we chose a well known algorithm from bioinformatics, the Smith-Waterman algorithm to demonstrate the speedups possible with FPGA acceleration. The Smith-Waterman algorithm has been accelerated usings FPGAs before by Compugen, TimeLogic and Starbridge, and using ASICs by Paracel.

2. The Cray XD1 Applications Acceleration Coprocessor

A unique feature of the Cray XD1 is the Application Acceleration Co-processor, shown in Figure 1.

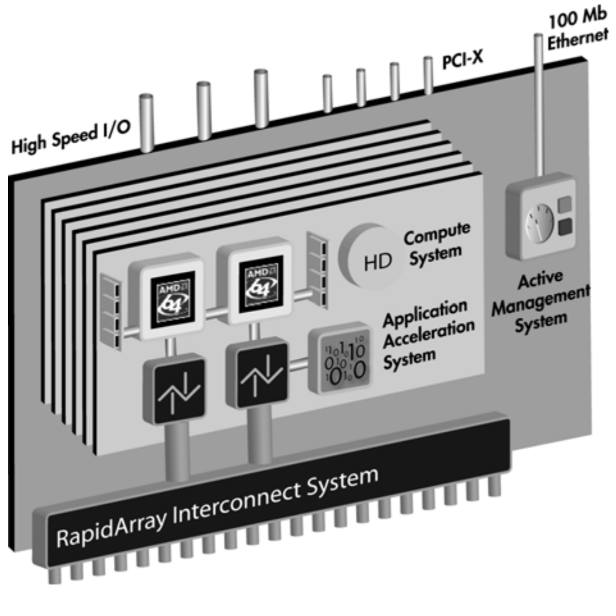

Figure 1. Cray XD1 Chassis with Application Accelerator

A Xilinx Virtex-II Pro VP50 FPGA is mounted on a daughterboard or “Expansion Module” on each SMP blade of an XD1 chassis. The FPGA chip is used as a coprocessor to the two Opterons in the SMP in order to speed up applications, hence the name. The expansion module contains 16 MB of high-speed QDR II static RAM as workspace for the FPGA.

The Expansion Module also contains a second Rapid Array Processor (RAP) to improve total system bandwidth. The FPGA is directly connected to the second RAP via a Hypertransport-like link called Rapid Transport (RT). This enables the FPGA to communicate with the local Opterons or the Rapid Array network with low latency and very high bandwidth- up to 3.2 Gbyte/sec bi-directional. This close coupling between the FPGA coprocessor and the host processors is very important when an application is split between the Opteron and coprocessor. In most typical systems the FPGAs are on a separate plug-in card on the PCI-X bus, and thus suffer increased latency and bandwidth restrictions.

When partitioning the application workload between the Opteron and FPGA, it is helpful to think of them as two different processor types working together. The Opteron is a fast, serial processor running at 2.4 GHz with 4-8 GB of RAM, optimised for floating-point work. The FPGA can be thought of as a slower but highly parallel processor running at 200 MHz with 16 MB of RAM, best suited for integer, logical and subword operations.

3. The Smith-Waterman Algorithm

The Smith Waterman dynamic programming algorithm is widely used in bioinformatics for biological sequence matching. It can be shown to give a provably optimum alignment, but it is an order of magnitude slower than heuristic methods such as BLAST. There is a rapid growth in the bioinformatics sequence databases and the scanning time to find sequence similarities is not getting shorter. The need for faster sequence scanning times and higher quality answers has fuelled the development of Smith Waterman hardware accelerators.

Dynamic programming is a method which uses nearest neighbour scores to compute a new score as sequence characters are evaluated over a database sequence. Higher scores identify goodness of sequence matches. The nature of the dynamic programming algorithm make it a good fit for fine grained parallelism using FPGAs to implement multiple processing elements (PEs).

Given two strings S1 and S2 of length s1 and s2 where s1>s2. We orient the shorter sequence S2 along the horizontal axis of scoring matrix S(y,x) and the longer sequence S1 along the vertical axis of scoring matrix S(y,x). The longer sequence is generally associated with a database sequence and the shorter sequence is generally associated as the target sequence. The scores matrix S(y,x) can be computed with the Smith Waterman dynamic programming algorithm described in the following equations, shown in Figure 2.

$$S(y,x) = \max\left\{\begin{array}{c} 0, \\ S(y-1,x-1) + Sub(cy,cx), \\ S(y,x-1) - eog, \\ S(y-1,x) - eog, \\ H(y,x-1) - e, \\ V(y-1,x) - e \end{array}\right\}$$

$$for\ 1 \le x \le s1, 1 \le y \le s2$$

$$where:$$

$$H(y,x) = \max\left\{\begin{array}{c} H(y,x-1) - e \\ S(y,x-1) - eog \end{array}\right\}\ for\ a\ given\ y$$

$$V(y,x) = \max\left\{\begin{array}{c} V(y-1,x) - e \\ S(y-1,x) - eog \end{array}\right\}\ for\ a\ given\ x$$

$$and:$$

$$S(0,x) = S(y,0) = 0$$

$$H(0) = V(0) = -eog$$

Figure 2. Smith-Waterman Equations

Sub(cy,cx) is a character substitution cost table that gives high scores for perfect match and lower scores for probable match. H(y,x) and V(y,x) are gap penalties associated with opening a gap and extending a gap in the sequence substitution. H(y,x) and V(y,x) is associated with opening gaps in the horizontal and vertical direction respectively. The value ‘eog’ is the cost of opening a gap and the value ‘e’ is the cost of extending a gap. Each value computed in S(y,x) come from one of 3 possible neighbour computations:

i) From the left or horizontal neighbour. This would open a gap in the string S1 along the vertical axis of S(y,x).

ii) From above or vertical neighbour. This would open a gap in the string S2 along the horizontal axis of S(y,x).

iii) From the upper left diagonal neighbour. This would mean a good match from character S1(y) and character S2(x).

The resulting S(y,x) give the similarity of the two segments of S1 and S2 in each (y,x) position. The sequence segment resulting in a similarity score in location S(y,x) can be determined by a traceback procedure which follows the path towards the origin S(0,0). For each S(y,x) position, traceback along the path to the neighbour where the S(y,x) score was derived. This process is repeated until we arrive a an S(y,x) value of 0. Depending on the path which was taken, each traceback step represents a gap along string S1, a gap along string S2, or a match in characters from S1 & S2. The traceback resulting from the largest value of S(y,x) is considered the most optimal match of subsequences in the two given sequences. When the derived score is equal from more than one neighbour, the traceback path is considered to be equivalent along any of those neighbours and the path choice is arbitrary.

An example with S1="ACGAACCCTTGC" and S2="ACGTATGC" is illustrated in Figure 3.

	0	A	C	G	T	A	T	G	C
0	0	0	0	0	0	0	0	0	0
A	0	2	0	0	0	2	0	0	0
C	0	0	4	2	1	0	1	0	2
G	0	0	2	6	4	3	2	3	1
A	0	2	1	4	5	6	4	3	2
A	0	2	1	3	3	7	5	4	3
C	0	2	4	2	2	5	6	4	6
C	0	0	2	3	1	4	4	5	6
C	0	0	2	1	2	3	3	3	7
T	0	0	0	1	3	2	5	3	5
T	0	0	0	0	3	2	4	4	4
G	0	0	0	2	1	2	2	6	4
C	0	0	2	0	1	0	1	4	8

Figure 3. Example of Smith-Waterman Algorithm

In the example of Figure 3, eog=2, e=1, match=2, mis-match=-1. The best similarity sequence score is 8 found in location S(12, 8) with the traceback subsequence shown in Figure 4.

```
A C G A A C C C T T G C
| | |   |         | | |
A C G T A - - - - T G C
```

Figure 4. ACGAACCCTTGC match to ACGTA----TGC

4.Implementation Decisions

When porting any algorithm to the XD1 with FPGA, decisions have to be made about how to partition the algorithm between the two processors. First, you have to identify code regions that are highly parallel, but small enough to fit multiple copies into the FPGA. You have to keep in mind that the FPGA has relatively limited local storage, though it is very fast. One of the most important considerations is the bandwidth between the FPGA and the Opteron. Though this is higher for the XD1 than for many other systems, it is possible to generate far more data on the FPGA than can be sent back over the links to Opteron memory.

Another important consideration is the data sizes for the internal and external data used by the FPGA. Since it is not restricted to any particular size, any number of bits may be used.

When porting the Smith-Waterman algorithm, it can be seen that filling the scoring matrix ("forward pass") is highly parallel, while the "traceback" is inherently serial. Filling the scoring matrix also represents the vast majority of the workload. It would then make sense to put the forward pass on the FPGA while putting the traceback and alignment on the Opteron. Unfortunately, bandwidth limitations prevent sending the entire scoring matrix back, since this would take longer than it took to calculate. The local memory of the FPGA would also limit how large a comparison could be done in a single pass. The matrix grows in size as the product of the length of the two strings, and can get quite large. It was decided to use the FPGA for scoring only, and for high-scoring matches that required an alignment the scoring matrix could be re-generated on the Opteron. This would seem inefficient, but in most library searches only one in 10,000 matches will need an alignment also.

For external data sizes we chose four bits for nucleotides and eight bits for amino acids. Internally, the scoring matrix is stored as 32 bits while the gap penalties and substitution matrix are stored as eight bits. The traceback information is stored in two bits.

5. Logic Design

We implemented the Smith-Waterman algorithm in an FPGA core called SWA, for Smith Waterman Accelerator. The SWA top level architecture consists of the RT Core, QDR II Core and a User Application block. The User Application Block contains all of the FPGA source code unique to the SWA application. The RT Core and QDR II core are standard IP blocks provided by Cray Inc. to enable communication with other devices over the fabric and access to the external QDR II RAMs.

The User Application Block consists of three sub-blocks: the RT Client, the Smith Waterman Processing Element (SWPE), and the Smith Waterman Control Element (SWCE). The RT Client provides a set of control and status registers for the design that allow an external device connected to the fabric (in this case the local node) to initialize and configure the SWA FPGA. The SWPE block computes a single similarity score and provides traceback information given two sequence characters. The SWCE controls the sequence characters and handles the transfer of computed information between the SWPE and memory.

The RT Core provides the fabric request interface which allows the local node to access the registers in the RT Client.

The following figure shows the architecture of the SWA FPGA design.

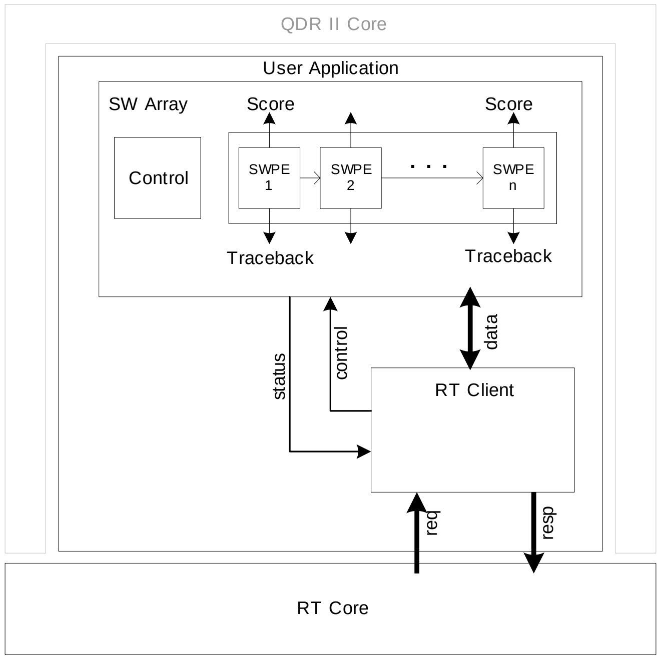

Figure 5. Architectural Block Diagram

The Smith Waterman Processing Element (SWPE) block implements the scoring function of the Smith Waterman algorithm. Each SWPE can compute one score for the scoring matrix. The SWPE is arranged in a linear unidirectional systolic array as shown in Figure 6.

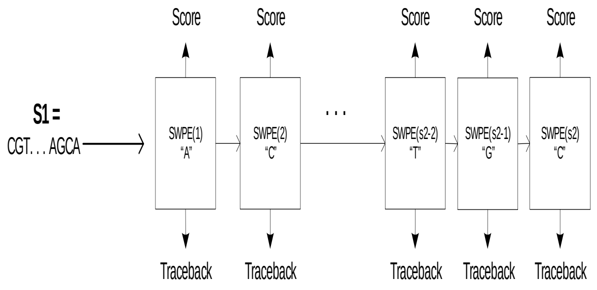

Figure 6. Smith-Waterman Systolic Array of Processing Elements

In Figure 6 database stream S1 is clocked into the SWPE one character at a time. When character S1(1) enters SWPE(1), the score and traceback information is computed for location S(1,1). The next computations occur when S1(1) enters SWPE(2) and S1(2) enters S1(1). The score and traceback information is computed for S(2,1) and S(1,2). This process continues until all characters of S1 enter SWPE(1) and exit SWPE(s2). The complete S(y,x) matrix is computed with elements along the minor diagonal computed in parallel as shown in Figure 7.

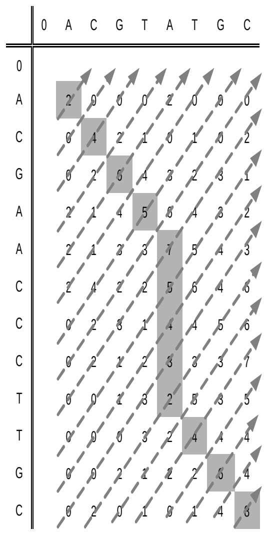

Figure 7. Parallel Computation of the Smith-Waterman Algorithm

The SWPE generates one 32 bit score and one 2 bit traceback per character comparison. Based on the number of SWPE which could fit on an FPGA, a large amount of data is generated in parallel. The RT interface can transfer slightly less than one quadword (64 bits) every 5 nanoseconds (200 MHz) to user ram. Therefore, more scoring matrix information is generated than the memory bandwidth to the user space. Even if the the traceback information only were sent back, this would become the dominant time usage in the algorithm. Therefore we decided to re-generate the matrix in Opteron memory in the (relatively rare) cases in which it was needed.

The actual SWPE block implementing the equations given in Figure 2 is shown below, on the next page.

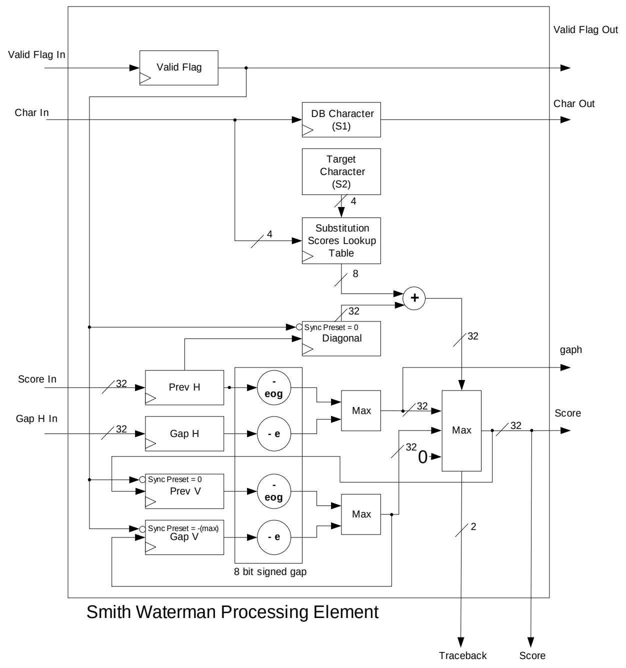

Figure 8. Smith-Waterman Processing Element

The SWPE block illustrated in Figure 8 shows all the components needed to implement the SW equations. The target sequence (S2) is expected to be shorter than the database (DB) sequence (S1). The SWPE is architected to have each character of the shorter sequence contained within the PE. The longer sequence is clocked through the all the PEs. This structure minimizes the number of times that the PEs need to be reloaded to extend the target sequence in situations where the target sequence is longer than the amount of PEs. The reloading process would require computations to stop and new target characters reprogrammed into the PE. The PEs are designed to have the target characters implemented as memory elements so that the FPGA configuration file does not need to be reloaded.

6. Performance Predictions

Performance of a Smith-Waterman algorithm is usually measured in Cell Updates Per Second (CUPS), or the number of cells in the scoring matrix that can be calculated per second. For our architecture, this rate can be calculated as:

Rate = FPGA freq. X clocks/cell X no. of SWPEs

For our current SWA core, the FPGA frequency is 80 MHZ, the number of clock cycles per cell update is 1, and the number of SWPES in the core is 32. This leads to a theoretical rate of 2.6 Billion CUPS. By overlapping computation and communications, we have been able to achieve 2.5 Billion CUPS on a real benchmark problem. For comparison, we ran the same benchmark problem on the Opteron only using the public-domain Smith-Waterman program SSEARCH34, part of the FASTA package. It achieved about 100 Million CUPS on the same benchmark. Thus, by using the FPGA we were able to achieve a 25-fold speedup over the Opteron alone.

The current core with 32 SWPEs only occupies about 60% of the FPGA, so it will be possible to fit more SWPEs on the current Virtex II Pro chip. However, a significant opportunity for speedup will occur in Q4 ’05, when the new Xilinx Virtex 4 FPGAs become available on the XD1. These chips contain approximately three times as many gates as the current Virtex IIs, and run at a somewhat increased clock frequency. With these new FPGAs it may well be possible to achieve 10 Billion CUPS or more.

7. Conclusions

The FPGA Application Acceleration Coprocessor on the Cray XD1 can offer a significant speedup over the Opteron processor alone on selected algorithms. It is necessary that the application have sufficient parallelism , proper workload distribution and appropriate data types for FPGA acceleration. A typical bioinformatics algorithm, Smith-Waterman, has shown these characteristics and many others are possible.

Careful attention must be paid to how the algorithm is partitioned between the Opteron and FPGA to achieve best use of the two processors. Bandwidth between the Opteron and the coprocessor is important, as well as appropriate choice of internal and external data sizes.

This new model of computing offers a real opportunity for high performance applications, particularly as newer and more capable FPGA chips become commercially available.

About the Authors

Jim Maltby is an applications engineer at the Seattle, Washington facility of Cray, Inc. **Steve Margerm** and **Jeffrey Chow** are hardware engineers at Cray Canada in Burnaby, BC.