High-performance hardware implementation of a parallel database search engine for real-time peptide mass fingerprinting

István A. Bogdán1, Jenny Rivers2, Robert J. Beynon2 and Daniel Coca1,∗

1Department of Automatic Control & Systems Engineering, The University of Sheffield, Mappin Street, Sheffield S1 3JD and 2Proteomics and Functional Genomics Group, Faculty of Veterinary Science, University of Liverpool, Crown Street, Liverpool L69 7ZJ, UK

Received on January 22, 2008; revised on April 1, 2008; accepted on April 29, 2008
Advance Access publication May 3, 2008
Associate Editor: John Quackenbush

ABSTRACT

Motivation: Peptide mass fingerprinting (PMF) is a method for protein identification in which a protein is fragmented by a defined cleavage protocol (usually proteolysis with trypsin), and the masses of these products constitute a ‘fingerprint’ that can be searched against theoretical fingerprints of all known proteins. In the first stage of PMF, the raw mass spectrometric data are processed to generate a peptide mass list. In the second stage this protein fingerprint is used to search a database of known proteins for the best protein match. Although current software solutions can typically deliver a match in a relatively short time, a system that can find a match in real time could change the way in which PMF is deployed and presented. In a paper published earlier we presented a hardware design of a raw mass spectra processor that, when implemented in Field Programmable Gate Array (FPGA) hardware, achieves almost 170-fold speed gain relative to a conventional software implementation running on a dual processor server. In this article we present a complementary hardware realization of a parallel database search engine that, when running on a Xilinx Virtex 2 FPGA at 100 MHz, delivers 1800-fold speed-up compared with an equivalent C software routine, running on a 3.06 GHz Xeon workstation. The inherent scalability of the design means that processing speed can be multiplied by deploying the design on multiple FPGAs. The database search processor and the mass spectra processor, running on a reconfigurable computing platform, provide a complete real-time PMF protein identification solution.

Contact: d.coca@sheffield.ac.uk

1 INTRODUCTION

Of the four key enabling tools and technologies in proteomics (protein separation, mass spectrometry instrumentation, protein databases and data processing, analysis and interpretation) it can be argued that the bioinformatics solutions lag in terms of performance and throughput. Post-instrument data processing in particular, is a major bottleneck in the proteomics workflow and as experimental design, instrument performance and user skills increase, it is expected this will become worse. A reasonable goal is that the processing and first-level (non-interpretable) analysis should be completed within the same timeframe as the experiment itself, preferably implemented as ‘near-instrument’ capabilities, where the user has the option of controlling the search parameters and strategies within the timeframe and constraints of the experiment. This largely rules out widely distributed multiprocessor computational farms/grid. However, computational platforms based on a relatively low number of conventional/multicore microprocessors are unlikely to deliver the speed required for real-time processing.

A solution to this problem, advocated in an earlier paper (Bogdan et al., 2007), is to implement proteomics data processing algorithms directly in hardware, effectively designing dedicated digital processors for every algorithm. Central to this approach is a rather unusual digital device termed a Field Programmable Gate Array (FPGA), which allows implementation and operation of a digital hardware design with the same facility as a conventional computer program. Early uses of FPGA devices in bio-computation were to accelerate gene sequence analysis (Fagin et al., 1993). FPGAs, which are well suited for high-performance, high-bandwidth and parallel processing applications, have been successfully employed to speed-up DNA sequencing algorithms (Guccione and Keller, 2002; Guerdoux-Jamet and Lavener, 1997; Hughey, 1996; Lavener, 1998; Simmler et al., 2004; Wozniak, 1997). FPGAs were also used in the attempt to accelerate search of substrings similar to a template in a proteome (Marongiu et al., 2003). More recently, FPGAs have been used to accelerate sequence database searches with MS/MS-derived query peptides (Anish et al., 2005). This hardware-based solution can locate a query within the human genome about 32 times faster than a software implementation running on a 2.4 GHz processor. A hardware sequence alignment tool implemented in FPGA is also available (Oliver et al., 2005). FPGA computing is used for comparing protein sequences with profile hidden Markov models (HMM) (Krogh et al., 1994; Sun and Buhler, 2007). An FPGA based BLAST search was used for expressed sequence tag (EST) sequencing (Panitz et al., 2007).

In succession to our earlier paper (Bogdan et al., 2007), the focus in the present article is to use FPGA-based computing to accelerate Peptide mass fingerprinting (PMF). PMF is a protein identification technique in which a protein is proteolyzed using an endopeptidase of defined specificity (usually trypsin) and the masses of the ensuing limit peptide fragments are measured. The proteins are identified by matching the measured molecular masses of these peptide fragments against theoretical peptide mass profiles generated from protein sequence database. PMF is readily delivered at high sensitivity through routine instrumentation such as
Parallel database search engine for real-time PMF

matrix-assisted laser desorption ionization time of flight (MALDI-TOF) mass spectrometers and although tandem MS approaches can recover more information from single peptides, PMF still plays an important role. Indeed, as more genomes are sequenced, and cross-species matching methods are developed, PMF may assume greater importance for many sub-proteome studies.

PMF involves two basic operations. The first is processing of the raw mass spectrum to derive a mass fingerprint, generating a dataset in which the only variable is the mass of each peptide (relative intensities of different ions are not routinely used in PMF). A hardware design of a raw mass spectrum processor that performs this operation was previously presented (Bogdan et al., 2007). When implemented in FPGA hardware, this solution delivered almost 170-fold speed-up compared to a conventional software implementation running on a dual-processor server.

This article addresses the second basic operation, which uses the peptide mass fingerprint to search protein databases for possible matches. Typically, a correlation score is computed between the database entries and the unknown peptide fragment mass list. The matches with the highest score are used to generate a candidate list of most likely proteins to have generated the PMF.

This article describes the design and hardware implementation of a highly parallel database search engine which, when searching the entire MSDB database, can generate the final candidate protein list in 240 ms. The design, which was implemented and run on a Xilinx XC2V8000 FPGA device and 1 GB of DDR SDRAM that can hold the entire MSDB database, can generate the final candidate protein list in 240 ms. Each FPGA module has one Virtex-II XC2V8000 FPGA at 100 MHz, achieves an average 1800-fold speed gain compared with an equivalent C software implementation, running on a single 3.06 GHz Xeon Server with 4 GB of memory.

2 METHODS

To create the raw data used to evaluate the FPGA implementation, single proteins were diluted with 50 mM ammonium bicarbonate and digested with trypsin at a protein to enzyme ratio of 50:1. Digestion was carried out at 37 °C for 24 h after which time, 1 μl of digested material was spotted onto a MALDI target. This was mixed with 1 μl a-cyano hydroxycinnamic acid matrix and analyzed using a Micromass M@LDI mass spectrometer (Waters, Manchester, UK) typically over the m/z range 800–4000.

The designs were implemented and tested on a modular and scalable reconfigurable hardware platform, consisting of a FPGA motherboard equipped with a Xilinx Virtex-II XC2V8000 FPGA (8 million gates) and 4 MB RAM, communicating with the host PC server via a peripheral component interconnect (PCI) interface. The motherboard has a Xilinx Virtex-II XC2V8000 FPGA, used to implement user designs—in our case the spectrum processor. A second, smaller Xilinx Spartan II FPGA implements the PCI interface protocols between the server PC and the FPGA system. Communication between these two FPGA devices is at 40 MHz on a 32 bit wide data bus. The motherboard can be configured to have up to three additional FPGA modules that can be plugged into dedicated motherboard slots (Fig. 1). At present, only one of these three modules has been used to implement the database search. Each FPGA module has one Virtex-II XC2V8000 FPGA device and 1 GB of DDR SDRAM that can hold the entire MSDB protein database (currently the encoded MSDB database is stored on a single module and takes about 680 MB). Each module is connected with the motherboard FPGA and with the other two modules via a 64 bit, 66 MHz local bus. This architecture enables the implementation of parallel searches at FPGA level as well as across modules.

The block diagram of the FPGA implemented database search engine is illustrated on Figure 2. In order to maximize database search speed, the search engine has been configured as a set of 48 identical search processors that can process database records (encoded protein streams) in parallel.

2.1 Database encoding and storage

In order to reduce communication overheads, the entire MSDB database was encoded and stored in the local 1 GB DDR SDRAM memory available on the FPGA module such that searching the database involves only on-board, local memory access (Fig. 1).

The MSDB database is available as plain ASCII text file. In practice, only 20 unique symbols are needed to encode all amino acids together with some additional standard symbols adopted in the FASTA format. Two additional symbols are used to indicate the end of a protein sequence and the end of the database. In total there are 28 symbols. These can be coded using only 5 bits (2^5 = 32 codes) instead of the minimum 8 required by the ASCII standard, which means that the effective size of the encoded database is about 40% smaller than the original.

The 1 GB DDR SDRAM memory available on the FPGA is organized in four banks with a 64-bit wide data bus each. The total data bus width is 256 bits. Each search processor has a 5-bit wide input, to receive a code-letter at each clock cycle from the database. Each memory module is able to supply 12 data streams of 5 bits synchronously to the corresponding 12 search processors that connect to the output of that RAM bank. The method allows division of the database into 4 x 12 = 48 data streams of consecutive records for parallel processing. Each data stream contains a variable number of complete protein sequences, the unused memory locations, which could not hold an entire protein sequence have been padded with zeroes (Fig. 3).

2.2 Database search processor

Each FPGA search processor performs the following basic operations:

- In silico protein digestion and peptide mass calculation according to externally specified digestion rules and post-translational modifications (PTM) (currently only fixed modifications implemented).
- Matching score calculation, based on pre-specified mass tolerance.

Search results consist of the indexes of the identified species in the database, the matched masses and the total score.
A match is found if

When a valid peptide fragment mass is compared in parallel with 13 mass/charge values (\(Y_i\)) stored in the database marker. The masses of individual amino acids, used to compute the peptide masses, are stored into a look-up table as 32 bits fixed-point numbers. Each search processor has three outputs: a processing end flag that remains set after processing ends until the search processor is reset; an output index that remains set to the last available FIFO address where the total number of matches is stored and a 38-bits output that contains the results (database index).

Figure 5 shows examples of cleavage site rules. The amino acid residues of the N-terminal side of the scissile bond are noted as P1, P2, …, P8 according to the Schechter and Berger nomenclature for the description of the protease subsites (Schechter et al., 1967). Cleavage site rules are given according to this notation.

The scoring unit counts the matches between the peptide mass fingerprint supplied externally and the peptide masses streamed out from the digestion unit. The precision of the comparison between the peptide mass fingerprint and the in silico peptide mass fingerprint is determined by the desired error tolerance (as p.p.m.) which may be dictated by the MS instrument accuracy. A peptide fragment mass fingerprint is compared in parallel with 13 mass/charge values (\(Y_i\)) and the in silico peptide mass fingerprint values \(Y_i\) can be specified by the user-defined parameter \(R\) that holds the desired error tolerance (as p.p.m.) which may be dictated by the MS instrument accuracy.

When a valid peptide fragment mass \(Y_i\) is computed, it is compared in parallel to the peptide mass fingerprint \((X_1, X_2, \ldots, X_n)\). In the current implementation \(n = 13\) so the computed peptide mass \(X_i\) is compared in parallel with 13 mass/charge values. The result of the comparison is used to generate the basic cumulative score for every processed peptide.

The number of \(n\) mass/charge values that are used in the search can, however, be increased at the expense of increasing the complexity of the design of individual search processors. This normally means that the number of processors that can be allocated on a single FPGA (currently 48) may need to be reduced. Since the search can be easily deployed on multiple FPGA modules, performance can be preserved or even expanded by adding additional modules. Alternatively, the user can decide the best tradeoff between performance and search strategy.

If a match is found, the score counter is incremented by one. The position of a match is also recorded in an \(n\)-bit (\(n = 13\) here) match index word. When the end of a record is found, the record index counter, the score counter and the match index register outputs are stored in intermediate registers.

If a peptide fragment mass fingerprint has multiple occurrences in a database record, it is found to match one of the mass/charge values from the input peptide mass fingerprint list, the score counter is incremented only once, after the first occurrence. A peptide fragment occurrence block is responsible for this function.

The basic matching score can be used to implement more sensitive searching schemes which account for peptide frequency distributions such as MOWSE (Pappin et al., 1993), PIUMS (Samuelsson et al., 2004) or more comprehensive Bayesian scoring approaches which also account for the individual properties of the proteins analyzed such as ProFound (Zhang et al., 1993). These scoring schemes have not yet been implemented in FPGA. Currently these scoring algorithms are run on the PC server. The externalization of the scoring statistics means that the output of the search can be rapidly evaluated using different scores, and even developed into a consensus score-validation scheme.
2.3 FPGA implementation
The actual database search design occupies about 99% of the FPGA's logic resources. 99% of the FPGA's internal RAM resources and 53% of the FPGA's I/O resources. The 1 GB on-board RAM is capable of a data rate of 100 MHz/cycle, which is the processing speed of the implemented database search engines. However the data transfers between the separate FPGA devices are at slower speed of 40 MHz. The design includes all necessary control and FPGA structures that implement a 64-bit wide data transfer between the FPGA devices. The FPGA board was installed and tested on Single and Dual 3.06 GHz Xeon processor servers with 4 GB RAM under Windows XP Professional.

All arithmetic operations on the m/z values were performed using 32-bit unsigned fixed-point binary number representation of mass and abundance values, with 12 bits after the radix point.

3 RESULTS

3.1 Speed gains
The MSDB database (August 31, 2006) was encoded and loaded in the local 1 GB DDR SDRAM module memory. The database contains 3,239,079 records with 1,079,594,700 effective code letters. If the additional separator codes are included the encoded database requires 3,239,079 bytes and occupies 67% of the available 1 GB.

A reference C program models the exact computational flow implemented by the hardware design. In tests, the output results of both the software and FPGA implementation of the database search engine are identical. The C program was run on Single and Dual 3.06 GHz Xeon PC servers under Windows XP Professional. In all simulations the FPGA implementation matched correctly the simulated peptide mass fingerprints.

The performance of both software and hardware (FPGA) designs were assessed using a randomly selected database records that were digested in situ using trypsin digestion rules. In each case, the search was carried out using 13 peptide m/z values selected randomly from the theoretical protein digests. The processing time for the software implementation accounts only for the main computational loop, after all variables have been initialized.

The FPGA system performs a complete database search in 240 (±0.02) ms while the completed average processing time for the C implementation is 7.2 min. As seen in Figure 6, the speed gain of the FPGA over the C software implementation ranges from 1650- to 1950-fold average speed-up.

The average processing time of the C implementation on a dual processor server is 6.8 min. The average speed gain in this case is 1695. It is worth noting that the search time quoted above could be reduced by pre-indexing the protein database—this strategy is employed by commercial software solutions. Processing time on FPGA is not dependent of the peptide mass fingerprint dataset and has linear dependency with the database size. To illustrate this aspect we have performed FPGA searches using only the database of Saccharomyces cerevisiae proteins. This database occupies only 0.277% of the RAM allocated for the full MSDB. In this case, the search time decreased to 0.66ms which represents 0.275% of the full database search time. In practice various subset databases can be loaded on-the-fly in the memory of the FPGA search module to carry out significantly faster single-species PMF searches, for example.

3.2 Experimental validation
We have tested the matching accuracy of the FPGA system using peptide mass fingerprints generated from raw MALDI-TOF mass spectra. Raw MALDI-TOF data were processed using the mass spectra processor detailed in Bogdan et al. (2007). The PMF consisted of m/z values selected from the identified peak list after the elimination of known contaminants such as trypsin and keratin.

In each case, the search was carried out using the FPGA system implementation and Mascot (www.matrixscience.com) with identical search parameters. In all cases, the same peptide matches were returned by both systems. For illustration, Table 1 shows the matching results from a batch of 10 mass spectra. Figure 7 shows examples of raw and processed mass spectra for an (chicken cardiac muscle). Table 2 shows the list of m/z values used in this particular search.

4 DISCUSSION
We have successfully demonstrated that an MSDB PMF database search can be implemented in FPGA. If the FPGA database search module is integrated with the FPGA spectrum processor described in Bogdan et al. (2007), the spectrum processing and database search time is around 241 ms. This means that the complete hardware PMF solution can process and match four mass spectra per second. This figure can be enhanced if the database search is distributed.
Table 2. Example of experimental PMF (actin) and the corresponding matched theoretical peptide masses

<table>
<thead>
<tr>
<th>PMF List (m/z)</th>
<th>Matched peptide masses by FPGA and Mascot</th>
<th>Error [p.p.m.]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1130.63</td>
<td>1130.54</td>
<td>79.6</td>
</tr>
<tr>
<td>1198.80</td>
<td>1198.70</td>
<td>83.4</td>
</tr>
<tr>
<td>1307.76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1354.74</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1364.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1374.86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1483.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1547.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1500.81</td>
<td>1500.70</td>
<td>73.3</td>
</tr>
<tr>
<td>1515.84</td>
<td>1515.74</td>
<td>66.0</td>
</tr>
<tr>
<td>1547.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1634.90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1791.03</td>
<td>1790.89</td>
<td>78.2</td>
</tr>
<tr>
<td>1956.14</td>
<td>1956.04</td>
<td>51.2</td>
</tr>
</tbody>
</table>

Tolerance was set at 250 p.p.m.

ACKNOWLEDGEMENTS

This work was funded by BBSRC (Grant No. BBS/B/16402). The authors gratefully acknowledge the support of Xilinx Inc. who donated the devices and design tools used in this study.

Conflict of Interest: none declared.

REFERENCES


