High Performance Computing from a General Formalism: Conformal Computing Techniques* Illustrated with a Quantum Computing Example

L. R. Mullin, Department of Computer Science
J. E. Raynolds, College of Nanoscale Science and Engineering
University at Albany, State University of New York, Albany, NY 12203

R. M. Mattheyses, GE Global Research, Niskayuna, NY, 12309

May 5, 2005

Abstract

As the theme of this conference suggests, developments in computer hardware have drastically outpaced developments in software. Software still suffers from a lack of standardization across architectures and processor/memory hierarchies. For example, despite the existence of language standards (i.e. ANSI) there are still no standards for compilers. The same source code produces different executable code even on the same hardware using different compilers. The formalism we use is based on a rigorous mathematical theory based on an algebra of abstract data structures (A Mathematics of Arrays) and an array indexing calculus (the psi-calculus). One of the most important aspects of this approach is the ability to compose a sequence of algebraic manipulations in terms of array shapes and abstract indexing. The net result is the elimination of temporary arrays, which leads to significant performance improvements. Another important point of this approach is that the mathematics used to describe the problem is the same as that used to describe the details of the hardware. Thus at the end of a derivation the resulting final expression can simply be translated into portable, efficient code in any programming language. Another important attribute of the Conformal Computing approach is the ability to mathematically prove that the resulting implementation is maximally efficient given a set of metrics (e.g. speeds of memory levels, processors, networks, etc). We choose to illustrate these techniques using a problem from Quantum Computing in which realistic simulations require enormous problem sizes. We present an algorithm that allows one to do a number of generalized matrix operations in a *single step* thus eliminating the need for large temporary arrays.

1 Quantum Computing: motivation for a matrix problem with arbitrary array access patterns

We now give a brief overview of the motivation for the present problem. The dream of Quantum Computing is the realization of a *quantum computer* in which data is represented by the states of a physical system such as the spin of an electron or proton. The physical picture one should imagine (to the extent that quantum processes can be imagined) is that of a spin or collection of spins interacting with electromagnetic fields. One possible embodyment of a quantum computer would be to utilize an apparatus closely resembling that used in Magnetic Resonance Imagining (MRI). Individual spins are manipulated through application of pulses of electric and magnetic fields.

The primary interest in Quantum Computing is the promise of greatly increased computing capability due to the inherently parallel nature of data storage and computation resulting from the superposition principle of quantum mechanics. For example, it has been theoretically proven that certain algorithms requiring exponential time on a classical computer can be solved in polynomial time on a quantum computer. We shall say no more about Quantum Computing in this paper and we turn to now the specific matrix problem to be solved.

2 The density matrix

Linear Algebra is the natural context in which to describe operations in a quantum computer and the central quantity is the density matrix. The density matrix contains time dependent information about the time evolution of the system. As such, when we want to describe the action of a particular field on a state of the system (e.g. flipping a spin with an RF electromagnetic pulse as in an MRI experiment), we write it as a matrix operation in which the density matrix acts on a vector. The vector in this case represents the state of the system before the application of the field. The structure of the matrix depends on the type of operation in question. In general, for any given operation, we require only a sparse collection of elements from the density matrix. The specific arrangement of the required elements in the matrix depends on which spin or collection of spins are being manipulated.

We focus on the following problem. Given the density matrix, for an arbitrary quantum operation on an arbitrary number of states (qubits) we wish (for computational convenience) to rearrange the data so as to place the required elements on the diagonal in block-diagonal form. Using the techniques of Conformal Computing we have found a way to do this *in one step*. Consider the following possible arrangements for a 16×16 density matrix for which we wish to manipulate two spins (qubits).

```
      0101010101010101010101010101010101010101010101010101010101010101
      0011001100110011001100110011001100110011001100110011001100110011
      0000111100001111000011110000111100001111000011110000111100001111
xxab  0000000010010100100110111001001011011011010111111
0000  a  b  c  d
0001  e  f  g  h
0010  i  j  k  l
0011  m  n  o  p
0100              a  b  c  d
0101              e  f  g  h
0110              i  j  k  l
0111              m  n  o  p
1000                        a  b  c  d
1001                        e  f  g  h
1010                        i  j  k  l
1011                        m  n  o  p
1100                                  a  b  c  d
1101                                  e  f  g  h
1110                                  i  j  k  l
1111                                  m  n  o  p
```

```
      0101010101010101010101010101010101010101010101010101010101010101
      0011001100110011001100110011001100110011001100110011001100110011
      0000111100001111000011110000111100001111000011110000111100001111
xaxb  0000000010010100100110111001001011011011010111111
0000  a  b        c  d
0001  e  f        g  h
0010        a  b        c  d
0011        e  f        g  h
0100  i  j        k  l
0101  m  n        o  p
0110        i  j        k  l
0111        m  n        o  p
1000                    a  b        c  d
1001                    e  f        g  h
1010                          a  b        c  d
1011                          e  f        g  h
1100                    i  j        k  l
1101                    m  n        o  p
1110                          i  j        k  l
1111                          m  n        o  p
```

We wish to rearrange a pattern such as the one on the right into a block-diagonal form such as that on the left. The transformation is effected by viewing the $2^n \times 2^n$ density matrix as a 2^{2n} dimensional hypercube and carrying out a certain rearrangement of the indices of the hypercube.

For a $2^n \times 2^n$ density matrix D, we say that the *shape* of the array (a vector giving the lengths of its dimensions) is $\rho D =< 2^n \ 2^n >$. Now we *reshape* the array into a hypercube D_h. Now the *shape* of the hypercube is a vector of 2^n 2's, that is, $<2\ 2\ \cdots 2>$.

The rearrangement we seek is a certain permutation of the indices of the hypercube. We write the block-diagonal matrix D_b as $D_b = \vec{t} \bigcirc\!\!\!\!\backslash \, D_h$, where the vector $\vec{t}$ is a permutation vector and the operator $\bigcirc\!\!\!\!\backslash$ corresponds to transposing the indices of the hypercube as specified by the permutation vector. For the specific example above, we have $\vec{t} =<0\ 2\ 1\ 3\ 4\ 6\ 5\ 7>$

We wave worked out an algorithm for determining the general permutation vector but space constraints prohibit us from presenting it here. Full details will be given during the presentation at hpec05.