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

Lenore R. Mullin
College of
Computing and Information

James E. Raynolds
College of
Nanoscale Science and Engineering

University at Albany, State University of New York

HPEC 2005

21 September, 2005

Overview

- **Conformal Computing: streamlining computation and shedding light on physics**
- **Breakthroughs obtained by restructuring *(reshaping)* multidimensional arrays to suit the problem and processor / memory / FPGA hierarchy**
- **Significant advances: FFT factors of 2 to 4 speedup**
- **Bit Reversal = *multi-dimensional* transpose**
  - » Fortran 95 definition is MoA definition
- **Fundamental view: The Hypercube**
- **This talk: Conformal Computing and Density Matrices**

Virtual Arrays

Connecting the Algorithm-Software-Hardware Boundary
(ideally the Physics-Algorithm-Software-Hardware)

- **Array *restructuring*: *reshape-transpose***
  - **An algebra of arrays and index calculus**
    - **MoA and Psi calculus:**
      - ***Conformal Computing***
    - ***Mullin-Raynolds Conjecture*:**
      - **Second Fundamental Theorem of the Psi Calculus: *Reshape-Transpose***
      - **First,** *is the* ***Psi Correspondence Theorem (PCT)***
        - **Mullin and Jenkins, Concurrency*: Practice and Experience 9-96***

Data Structure *insights*

- The *structure* of density matrices
  - What is the *structure* really?
  - Is the *matrix* the ideal way of seeing a quantum algorithm?
    - Are there other representations more ideal?
    - Must we always use Permutation Matrices to permute indices?
    - Can we envision a quantum algorithm?
      - Hypercubes: $\mathcal{L}^2$ space

Reshape-Transpose

- Array *restructuring: reshape-transpose*
  - Restructure the density matrix
  - Restructure to lift dimension to match processor / memory / FPGA hierarchy.
  - View qubits as coordinates in a hyperspace
    - *Reshape-transpose* and *hypercube* common themes in FFT:
      - *bit reversal* is *hypercube transpose*
      - *transpose vector* to define *butterfly in FFT*
      - *transpose vector* to define *cache loop in FFT*
        - » Computer Physics Communications
        - » Materials Research Society
        - » Digital Signal Processing
  - *NO Permutation Matrices and NO Matrix Multiplication to permute indices*

Example: *Block Decomposition*

$$A = \left[\begin{array}{cc|cc} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ \hline 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{array}\right]$$

2-dimensional

Viewed as
4-dimensional

$$A'' = \langle 0\,2\,1\,3 \rangle \phi A' = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 0 & 1 \\ 4 & 5 \end{bmatrix} \\ \begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 8 & 9 \\ 12 & 13 \end{bmatrix} \\ \begin{bmatrix} 10 & 11 \\ 14 & 15 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Array *“shapes”*

- ***Shape operator:*** ρ **returns a vector containing the lengths of each dimension**
- **Total number of components in *A* is *16.***

$$\rho A = \langle 4\ 4 \rangle$$

- ***Shape of A (two-dimensional):***

$$\rho A' = \langle 2\ 2\ 2\ 2 \rangle$$

- ***Shape of A’ (four-dimensional):***

- ***Shapes are factors* of the total number of components.**
- ***Factors* fit the physics and *factors* fit the levels of processor/memory/FPGA/…**

“Reshape” Operator

$$A = \left[\begin{array}{cc|cc} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ \hline 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{array}\right]$$

2-dimensional

The process of *“lifting”* the dimension is carried out with the *“reshape”* operator

$$A' = \langle 2\ 2\ 2\ 2 \rangle \hat{\rho}\ A$$

Becomes
4-dimensional

$$A' = \left[\begin{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \\ \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 8 & 9 \\ 10 & 11 \end{bmatrix} \\ \begin{bmatrix} 12 & 13 \\ 14 & 15 \end{bmatrix} \end{bmatrix} \right]$$

“Transpose” operator

- ***Transpose*** **operator ϕ permutes the dimensions**

$$A' = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \\ \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 8 & 9 \\ 10 & 11 \end{bmatrix} \\ \begin{bmatrix} 12 & 13 \\ 14 & 15 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

transpose vector

$$A'' = \underset{\uparrow}{\langle 0\,2\,1\,3 \rangle} \phi A' = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 0 & 1 \\ 4 & 5 \end{bmatrix} \\ \begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 8 & 9 \\ 12 & 13 \end{bmatrix} \\ \begin{bmatrix} 10 & 11 \\ 14 & 15 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

“Hypercube” Representation

- The arrays A' and A'' are examples of “*hypercubes*”
  - » multi-dimensional *unit-cubes*
- Often array operations simplify in the hypercube representation, *e.g. bit reversal, permutations*
- In a hypercube: all dimensions have *length 2*
- A hypercube allows the input vector to be viewed in the *highest dimension possible*.
- Across dimensions every component can be related to every other component, i.e. permutations are easily made.

xxab	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
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

	00	01	10	11
00	a	b	c	d
01	e	f	g	h
10	I	j	k	l
11	m	n	o	p

The punch line

- Through direct indexing, arbitrary data re-arrangements can be performed in ONE STEP
- This leads to exceedingly efficient computation
- Fundamental perspective: by viewing the data in computation in the most general way is leading to new insights into the underlying physics
- Notice ALSO: all the squares on the diagonal can be accessed in *parallel*, I.e. over the primary axis index or processor index(or cache index or whatever we are using).

Computation and Gates

- From Classical to Quantum Gates
  - Classical XOR
    - Diagram
    - Boolean Algebra
    - Logic Expression: Boolean Table
  - Reversible XOR: Controlled NOT
    - Diagram
    - Boolean Table
  - Quantum NOT: Reversible
    - Linear Algebra

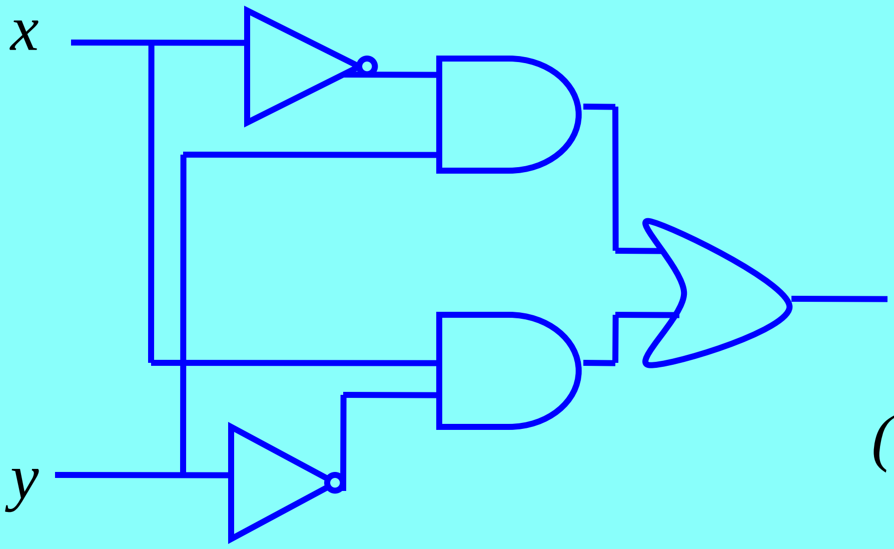
From *Classical* to *Quantum* Computing

- Basic Gates in Classical Computers
    - and, or, not
- Basic Gates in Quantum Computers
    - not, controlled not, controlled- controlled not

Major Differences

- ONE state versus ALL states
- Boolean Algebra versus Linear Algebra
- Irreversible versus Reversible

  computation



Classical *xor* **2 bits in 1 out**

x	y	*xor(x,y)*
0	0	0
1	0	1
0	1	1
1	1	0

$$(\overline{x} \circ y) + (\overline{y} \circ x)$$

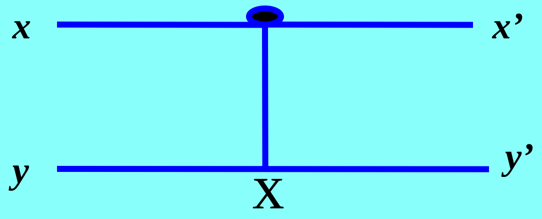
$$((x=0)\&(y=1)) \mid ((y=0)\&(x=1))$$

Reversible *xor* **2 bits in 2 out**

Controlled NOT (classical implementation)

cnot(x,y)

x	y	x’	y’
0	0	0	0
1	0	1	1
0	1	0	1
1	1	1	0



Some Notation

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \quad \text{or} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

- Use *matrices* to denote states
  - •The above are *basis states* in an abstract space (Hilbert space).
- Linear Algebra to relate gate operations
- Classical states use Boolean Algebra

Basis states: what are they, really?

- Physical example: the states $|0\rangle$ and $|1\rangle$ can be realized as the ***spin-down** and **spin-up*** states of a spin-1/2 particle such as an electron.
- States (information) are manipulated through the application of electro-magnetic fields.
- Example: application of an EM pulse can flip a state from down to up (just like in Nuclear Magnetic Resonance spectroscopy).

$$|\Psi(t)\rangle = \alpha(t)|0\rangle + \beta(t)|1\rangle$$

$$\alpha(0)=1; \beta(0)=0 \;\Rightarrow\; \alpha(\tau)=0; \beta(\tau)=1$$

Some Notation (cont.)

- **General state: superposition (linear combination)**

$$\alpha|0\rangle + \beta|1\rangle = \alpha\begin{bmatrix}1\\0\end{bmatrix} + \beta\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha\\0\end{bmatrix} + \begin{bmatrix}0\\\beta\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\end{bmatrix}$$

$$\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \times \begin{bmatrix}\alpha\\\beta\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\end{bmatrix}$$

$$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix} \times \begin{bmatrix}\alpha\\\beta\end{bmatrix} = \begin{bmatrix}\beta\\\alpha\end{bmatrix}$$

Higher Dimensions

- Basis states in higher dimensions from Cartesian products of $|0\rangle$ and $|1\rangle$

- For example

$$|00\rangle = |0\rangle|0\rangle$$

$$|01\rangle = |0\rangle|1\rangle$$

$$|10\rangle = |1\rangle|0\rangle$$

$$|11\rangle = |1\rangle|1\rangle$$

- A general state is a linear combination of these basis states in this 4-dimensional space:

Example: CNOT *(controlled not)*

$$\Psi = (\alpha|0> + \beta|1>)|0> = \alpha|00> + \beta|10> =$$

$$\Psi = \begin{pmatrix} \alpha|00> \\ 0|01> \\ \beta|10> \\ 0|11> \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \\ \beta \\ 0 \end{pmatrix}$$

$$\text{CNOT} = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \end{pmatrix}$$

$$\text{CNOT} \quad \Psi = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \end{pmatrix} \begin{pmatrix} \alpha \\ 0 \\ \beta \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \\ 0 \\ \beta \end{pmatrix} = \alpha|00> + \beta|11>$$

Physical Observables

- **Measurable quantities** $A(t)$ **calculated as averages of operators:** $\hat{A}(t)$

- **Wave function vs. density matrix representation**

$$A(t) = \langle \psi(t) | \hat{A}(t) | \psi(t) \rangle = Tr[\hat{\rho}(t)\hat{A}(t)]$$

$$Tr[\hat{a}\hat{b}] = \sum_{lm} a_{lm} b_{ml}$$

Quantum Simulation

- Quantum simulators: *we can't build many qubit quantum computers YET*
  - One method: density matrix method
    - Computations are Gate Operations
    - Gate Operations are Matrix Operations
    - Linear Algebra
    - Algebra of Arrays(MoA): Algorithm and Architecture

Industrial applications

- GE-Lockheed Martin Objectives
  - Flexible extensible simulator for quantum algorithms
  - Provide high performance throughput
    - Advanced Architectures: NEED portable, scalable designs, optimal performance.
      - May include multiple processors, levels of memory, FPGAs.
        - SGI MOATB
        - Cray XD1
    - Exploit sparseness and structure of gate operators
    - Simulate systems with more than 14 qubits
    - In a Quantum Computer we require in excess of 2^{31} bytes

Simulations

Why simulate?

Quantum computers are difficult to build

- **Usually small laboratory experiments: 4-5 qubits**

Major error mechanisms can be modeled

- **Hardware imperfections and physical phenomena**

Simulation allows observation of intermediate states

- **Reversible conventional gates**

Use the simulator to explore *quantum algorithm* development for *digital and image processing* applications, e.g. *FFT*

abxx	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	a				b				c				d			
0001		a				b				c				d		
0010			a				b				c				d	
0011				a				b				c				d
0100	e				f				g				h			
0101		e				f				g				h		
0110			e				f				g				h	
0111				e				f				g				h
1000	I				j				k				l			
1001		I				j				k				l		
1010			I				j				k				l	
1011				I				j				k				l
1100	m				n				o				p			
1101		m				n				o				p		
1110			m				n				o				p	
1111				m				n				o				p

	00	01	10	11
00	a	b	c	d
01	e	f	g	h
10	I	j	k	l
11	m	n	o	p

Density Matrix $2^n \times 2^n$ to Quantum Density Hypercube

2-d to *2n-d*

• one qubit:	2 by 2 matrix
• two qubits:	4 by 4 matrix
• three qubits:	8 by 8 matrix
• …	

Goal: Create a **Quantum Algorithm** to perform ***n* qubit-gate** operations that is **true** to the **physics** AND **computational platform.** ***Thus, all designs are verifiable, and scalable to existing AND emerging architectures.***

Density Matrix $2^n \times 2^n$ to Quantum Density Hypercube

2*-*d* to *2n*-*d

- **View the *qubits* as *coordinates* in the Quantum Density Hypercube**
- **Create a *permutation vector* that will be used to perform a *multi-dimensional* transpose on the Quantum Density Hypercube.**
- **The result of this transpose aligns matrices on the *diagonal* of the original Density Matrix**

- **Gate is then applied**, again noting the gates can be applied in parallel , i.e. the processor index is the primary axis index.

Density Matrix$_{2^n \times 2^n}$ to Quantum Density Hypercube

2-d **to** ***2n*-d**

- **The *design* and subsequent *implementation* uses the least amount of resources.**

- ***Normal forms*** **after MoA and Psi Analysis, yield a *generic design* independent of platform.**

Qubits, Indices, and Permutations

Example: 16 by 16 Density Matrix becomes a
2^8 Quantum Density Hypercube

Note that the design is for 2^n by 2^n , $0<=n$, n in I^+
AND *any number of bits*.

Recall the xls file previously shown

Qubits, Indices, and Permutations

Assumptions in the Example:

- Let *a* and *b* denote which bits to gate*:*
  - •*xxab: bits 0 and 1* *axxb: bits 0 and 3 …*
- *bits* are numbered from *right* to left*:*
  - *1 1 1 0* is used to evaluate its decimal equivalent

    $(1 * 2^{3}) + (1 * 2^{2}) + (1 * 2^{1}) + (0 * 2^{0})$
- *indexing* is numbered from *left* to right
  - As a vector*,* < *1 1 1 0* > when indexed would yield*:*

    < 1 1 1 0 >[0] = 1

    …

    < 1 1 1 0 >[3] = 0

Qubits, Indices, and Permutations

Example(cont.): From qubits to permutation vector

- bits 0,2: xaxb -> 3 2 1 0 3 2 1 0 bit ordering
  0 1 2 3 0 1 2 3 index ordering
  0 2 1 3 0 2 1 3 bit 2 is index 3
  swap bits 1 and 2
  <0 2 1 3 4 6 5 7> is the transpose vector

- bits 1,2: xabx -> 3 2 1 0 3 2 1 0 bit ordering
  0 1 2 3 0 1 2 3 index ordering
  0 1 3 2 0 1 3 2 swap bits 2 and 3
  0 3 1 2 0 3 1 2 swap bits 1 and 2
  <0 3 1 2 4 7 5 6> is the transpose vector

- bits 0,3: axxb -> <2 1 0 3 6 5 4 7>
- bits 1,3: axbx -> <3 1 0 2 7 5 4 6>
- bits 2,3: abxx -> <2 3 0 1 6 7 4 5>

From Permutation Vector to the Diagonal

Given: a permutation vector denoted by $\vec{t}$, ***the transpose vector***, permute all indices.

- **Apply binary transpose** after **reshaping**(restructuring) the **density matrix** into a **density hypercube**.
  - ➢ Permute all indices as defined by the **transpose vector**
- **Gated arrays** are now **on the diagonal**.

From Permutation Vector to the Diagonal

- Indices are calculated and addressed directly from the original array stored in memory:
  - ➢Algebraically the Physics
  - ➢Algebraically all at once
  - ➢Algebraically decomposable to present and future architectural platforms(even quantum)
  - ➢Algebra remains the same throughout
    - ➢the problem,
    - ➢the decomposition over processor/memory/FPGA,
    - ➢the mapping,
    - ➢the architectural abstraction,
    - ➢verifiable designs

The General Expression
Moa and Psi Reduction

Given DM s.t. ρDM = $< 2^n\ 2^n >$, restructure to a hypercube, QDH.

Let $\vec{s}$ denote the shape of QDH s.t. $\vec{s} = < 2n\ \ \rho\hat{}\ \ 2 >$

$$= \underbrace{< 2 \ldots 2\ 2 >}_{2n}$$

Then QDH = $\vec{s}\ \rho\hat{}$ DM

Use $\vec{t}$, the transpose vector previously defined.

Perform the binary transpose:

$$\vec{t}\ \ \bigcirc\!\!\!\!\backslash\ \ \text{QDH}$$

Now, all matrices defined by bits chosen are on the diagonal.

Note: Restructuring back to DM and indexing creates no new arrays because of *Psi Reduction*.

The General Expression
Moa and Psi Reduction

$$\vec{t} \;\cancel{\bigcirc}\; \text{QDH}$$

- This expression moves all gated coordinates to the diagonal (after reshaping) of DM.
- This expression describes the Physics in one operation.
- Now Psi Reduce to normal form -> Generic Design.

$$\text{forall } i \text{ s.t. } 0 <= i < 2^{2n}$$

$$\vec{i} = \gamma'(\vec{i} ; s)$$

$$\vec{i} = \vec{i}[\vec{t}]$$

$$@DM + \gamma(\vec{i} ; \vec{s})$$

This is the *Generic Normal Form*

Conclusions

- **MoA** and **Psi Calculus** *quantum algorithm* for **density matrix optimizations.**
  - Describes the **physics naturally.**
  - qubit access and gate application.
    - **Independent** of number of qubits.
    - **Independent** of density matrix size.
  - Describes **decomposition** and **mapping**.
    - Multiple processor/memory levels.
      - Normal form is a **generic design** independent of target architecture, ideally the physics directly.

The punch line

- Through direct indexing, arbitrary data re-arrangements can be performed in ONE STEP
- This leads to exceedingly efficient computation
- Fundamental perspective: by viewing the data in computation in the most general way is leading to new insights into the underlying physics

Future Directions

- **Fundamental concepts:** ***reshape-transpose, hypercube representation…just beginning to be explored***

- **Connections with quantum algorithms: Quantum FFT, Shor's factoring algorithm, etc**

- **Highly efficient practical designs for today's (and tomorrow's) computers!**

Acknowledgement

- We thank Robert Mattheyses from GE Global Research for introducing us to this problem.

Thank you